UNIVERSITAS INDONESIA

ANALISIS KINERJA PEMILIHAN FITUR UNTUK *SUPPORT VECTOR MACHINE* (SVM) PADA MASALAH ANALISIS SENTIMEN

SKRIPSI

ADHIMAS YUDHA PRAWIRA
1006673172

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM SARJANA MATEMATIKA
DEPOK
JUNI 2014

UNIVERSITAS INDONESIA

ANALISIS KINERJA PEMILIHAN FITUR UNTUK *SUPPORT VECTOR MACHINE* (SVM) PADA MASALAH ANALISIS SENTIMEN

SKRIPSI

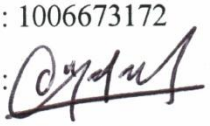
Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

ADHIMAS YUDHA PRAWIRA
1006673172

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM SARJANA MATEMATIKA
DEPOK
JUNI 2014

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	: Adhimas Yudha Prawira
NPM	: 1006673172
Tanda Tangan	: 
Tanggal	: 18 Juni 2014

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama	:	Adhimas Yudha Prawira
NPM	:	1006673172
Program Studi	:	Sarjana Matematika
Judul Skripsi	:	Analisis Kinerja Pemilihan Fitur untuk *Support Vector Machine* (SVM) pada Masalah Analisis Sentimen.

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Dr. rer. nat. Hendri Murfi, S.Si, M.Kom	(.....)
Penguji	: Dr. Gatot F. Hertono, Ph.D	(.....)
Penguji	: Dra. Siti Aminah, M.Kom	(.....)

Ditetapkan di : Depok

Tanggal : 18 Juni 2014

KATA PENGANTAR

Alhamdulillahirabbil ‘alamiin,

Segala puji bagi Allah SWT yang telah memberikan rahmat dan ridho-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai salah satu hasil proses pembelajaran penulis dalam rangka menuntut ilmu di departemen Matematika UI. Terimakasih kepada Ayah, Ibu, dan adik-adik penulis yang selalu memberikan kasih sayang, doa, serta segala dukungan selama ini kepada penulis. Penulis juga menyadari bahwa tanpa bantuan berbagai pihak, penulis akan mengalami banyak kesulitan dalam menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terimakasih kepada:

1. Dr. rer. nat. Hendri Murfi, S.Si, M.Kom selaku pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk membimbing penulis dalam penyusunan skripsi ini. Terima kasih atas segala kesabaran, nasehat dan pembelajaran yang telah diberikan.
2. Dra. Denny Riama Silaban, M.Kom selaku pembimbing akademik.
3. Bapak dan Ibu pengajar departemen Matematika atas segala ilmu dan nasehat yang telah diberikan.
4. Seluruh staf dan karyawan Departemen Matematika UI atas semua bantuan yang telah diberikan.
5. Pino Rachmandika, Eka Widowati, Tasya Rahmita, serta Abdul Choliq Ad Dakhil yang telah meluangkan waktunya untuk mendengarkan cerita-cerita dan keluh kesah penulis. Terimakasih atas seluruh waktu, bantuan, canda tawa, kebersamaan, nasehat, dan semangat tiada batas yang telah diberikan kepada penulis.
6. Seluruh teman-teman satu bimbingan skripsi.
7. Seluruh keluarga Matematika 2010 yang telah memberikan warna dalam kehidupan penulis selama di Matematika UI.
8. Bangkit, Ardhian, Fatkhan, Mas Mivtah, Icho – Riza, Kholid, Jodi, Anang, Grafika, dan seluruh teman-teman kos maupun kontrakan penulis. Terimakasih atas *sharing* ilmu, nasehat, dan suka – duka di perantauan.

9. Seluruh keluarga Matematika angkatan 2007, 2008, 2009, 2011, 2012, 2013 atas seluruh pembelajaran bersama di Matematika.
10. Seluruh teman-teman seperjuangan dari Kompor UI. Terimakasih atas segala bantuan dan *sharing* pengalamannya selama di UI.
11. Ibu Is dan seluruh teman-teman KOMA (Yudhis, Yono, Gabe).
12. Seluruh siswa STKQ maupun Pesma Al-Hikam atas segala doa dan semangatnya.
13. Seluruh pihak yang telah membantu penulis baik langsung maupun tak langsung yang tidak dapat penulis sebutkan namanya satu-persatu disini.

Semoga Allah SWT berkenan membalas segala kebaikan pihak-pihak yang telah membantu.

Penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan. Maka dari itu, penulis mohon maaf jika masih terdapat kesalahan maupun kekurangan dalam penulisan skripsi ini. Penulis juga menerima segala kritik dan saran terhadap skripsi ini. Semoga skripsi ini bermanfaat bagi pengembangan ilmu pengetahuan dan kemajuan bangsa Indonesia.

"Seperti halnya *machine learning*, semakin banyak dan beragam pengalaman yang kamu peroleh akan membuat kamu lebih bijaksana."

Juni 2014

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Adhimas Yudha Prawira
NPM	: 1006673172
Program Studi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

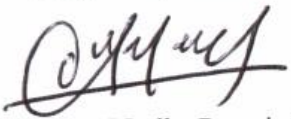
demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti *Non*eksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Analisis Kinerja Pemilihan Fitur untuk *Support Vector Machine* (SVM) pada Masalah Analisis Sentimen

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti *Non*eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok
Pada tanggal: 18 Juni 2014
Yang menyatakan



(Adhimas Yudha Prawira)

ABSTRAK

Nama : Adhimas Yudha Prawira
Program Studi : Matematika

Judul : Analisis Kinerja Pemilihan Fitur untuk *Support Vector Machine* (SVM) pada Masalah Analisis Sentimen

Twitter merupakan salah satu media sosial yang digunakan secara *massive* di Indonesia. Para pengguna Twitter ini membicarakan berbagai macam hal, salah satunya terkait pencalonan presiden. Perbincangan para pengguna Twitter ini memiliki nilai sentimen baik positif maupun negatif. Dukungan masyarakat terhadap masing-masing kandidat calon presiden dapat diketahui dengan melihat sentimen masyarakat melalui perbincangan mereka di Twitter, hal ini sering disebut juga sebagai analisis sentimen. Namun, jumlah pengguna dan obrolan para pengguna Twitter yang sangat banyak mengakibatkan data yang akan diproses membutuhkan waktu yang cukup lama. Untuk melakukan proses analisis sentimen para pengguna Twitter secara cepat dan otomatis dapat digunakan bantuan mesin. Salah satu metode yang digunakan untuk melakukan proses analisis sentimen adalah *Support Vector Machine* (SVM). Pada dasarnya, semakin banyak data yang digunakan sebagai *data training* dalam pemilihan model fungsi klasifikator maka akan memberikan generalisasi akurasi analisis sentimen untuk *data testing* yang tinggi pula. Namun di sisi lain, semakin banyaknya *data training* juga akan menyebabkan besarnya dimensi ruang fitur. Hal ini membuat mesin membutuhkan waktu yang cukup lama dalam melakukan pembentukan fungsi klasifikator. Untuk menanggulangi hal ini, akan dilakukan metode optimasi fitur sehingga mesin dapat tetap membentuk fungsi klasifikator dengan akurasi yang tinggi namun dengan dimensi ruang fitur yang rendah.

Kata Kunci : Analisis Sentimen, Optimasi Fitur, *Supervised Learning*, *Support Vector Machine* (SVM), Twitter.
xiii + 52 halaman : 17 gambar dan 7 tabel
Daftar Pustaka : 15 (1997-2013)

ABSTRACT

Name	:	Adhimas Yudha Prawira
Study Program	:	Mathematics
Title	:	Performance Analysis of Feature Optimization for Support Vector Machine (SVM) on Sentiment Analysis Problems

Twitter is a social media that used in Indonesia massively. Twitter users talk (tweet) about various things, one of them is about presidential nomination. Twitter user conversations have a positive or negative sentiment. Community support for each presidential candidate can be determined by looking at the public sentiment through their conversations on Twitter, this is often referred to sentiment analysis. However, the number of users and tweets cause the data to be processed requires quite a long time. Machine can be used to make the process of Twitter sentiment analysis quickly and automatically. One method that used to perform the sentiment analysis process is a Support Vector Machine (SVM). Basically, the more data that used as data training in the model selection function will give a high accuracy generalization sentiment analysis on data testing. On the other hand, the increasing number of training data will also cause large dimensional feature space. This makes the machine takes a long time to perform model selection. To overcome this problem, feature optimization will be performed. Feature optimization will preserve the high accuracy of the model, but with a low dimensional feature space.

Keywords	:	Feature Optimization, Sentiment Analysis, Supervised Learning, Support Vector Machine (SVM), Twitter.
xiii + 52 pages	:	17 pictures and 7 tables
Bibliography	:	15 (1997-2013)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

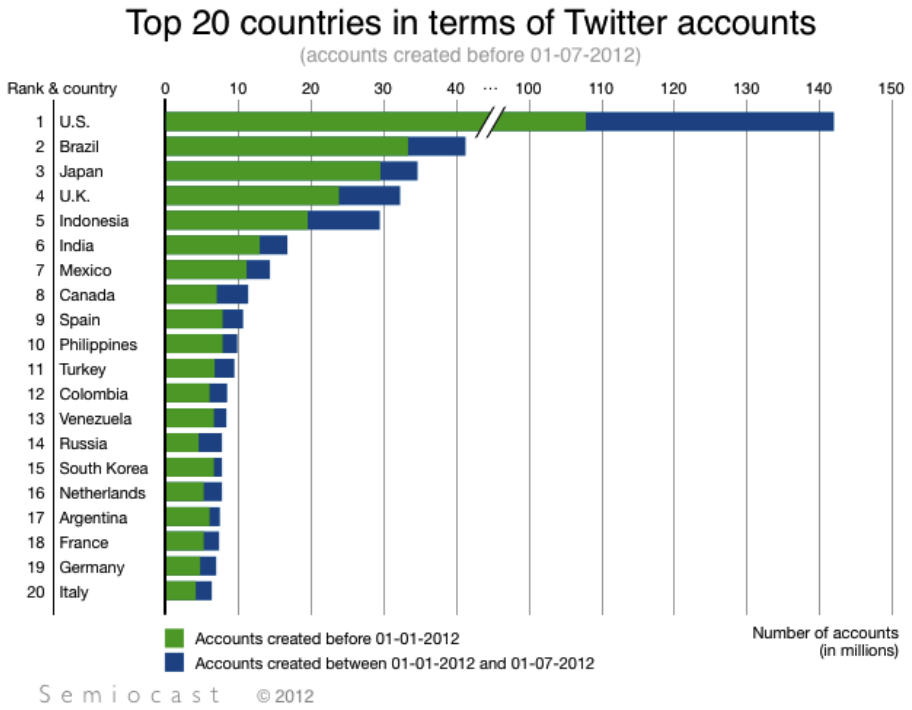
Indonesia merupakan suatu negara kesatuan yang dipimpin oleh seorang presiden. Setiap lima tahun sekali, pemilihan umum diadakan untuk memilih presiden dan wakil presiden untuk jabatan selama lima tahun ke depan. Menjelang pemilu, para calon presiden berusaha mempromosikan diri melalui berbagai media. Hal ini membuat masyarakat memberikan perhatian dan saling bertukar opini tentang masing-masing kandidat calon presiden.

Menjelang pemilu, banyak lembaga masyarakat yang melakukan *survey* untuk mengetahui seberapa besar dukungan masyarakat terhadap para calon presiden. Namun, *survey* ini sebagian besar masih dilakukan secara manual dengan mendatangi sebagian penduduk untuk pengambilan sampel.

Di sisi lain, perkembangan teknologi informasi khususnya internet di Indonesia sudah sangat cepat. Munculnya berbagai media sosial di dunia termasuk di Indonesia juga memicu cepatnya perkembangan arus informasi. Salah satu media sosial yang memicu hal tersebut adalah Twitter. Twitter merupakan salah satu media sosial yang digunakan secara *massive* di Indonesia. Menurut data dari *semiocast.com* pada tahun 2012, Indonesia merupakan negara dengan pengguna Twitter terbesar ke-lima di dunia[1]. Selain itu, Twitter juga merupakan salah satu media sosial yang kerap digunakan oleh masyarakat dalam mengungkapkan opini mereka berkaitan dengan hal apapun, termasuk calon presiden. Oleh karena itu, perbincangan masyarakat dalam Twitter (disebut *tweet*) berkaitan dengan calon presiden merupakan suatu data yang cukup relevan untuk menggambarkan dukungan masyarakat kepada masing-masing calon presiden.

[1] Dipublikasikan di http://semiocast.com/, diakses pada 2 Januari 2014 pukul 08.00 WIB.

Dengan banyaknya *tweet* yang dihasilkan dan banyaknya pengguna Twitter di Indonesia akan mengakibatkan data yang dihasilkan sangat besar. Untuk mengolah dan melakukan analisis sentimen opini masyarakat dari data tersebut secara manual dibutuhkan waktu yang tidak singkat. Maka dari itu bantuan mesin diperlukan untuk mengklasifikasikan *tweet* yang bernilai sentimen setuju maupun tidak setuju dalam waktu yang relatif singkat dan akurat yang selanjutnya disebut analisis sentimen. Salah satu metode yang dapat digunakan untuk melakukan klasifikasi dalam analisis sentimen adalah metode *Support Vector Machine* (SVM).



Gambar 1. 1 Grafik pengguna Twitter di dunia

[Sumber: http://semiocast.com, waktu akses: 08.00 WIB, 2 Januari 2014.]

Pada dasarnya, semakin banyak *data training* yang diberikan kepada mesin, maka akurasi model fungsi klasifikator yang dibentuk oleh mesin juga semakin tinggi. Namun dalam melakukan representasi ke dalam vektor numerik, dimensi data menjadi besar yang dikarenakan oleh banyaknya fitur. Optimasi fitur perlu dilakukan terhadap *data training* dengan memperkecil dimensi *data training* namun tetap mempertahankan akurasi model yang tinggi. Faktorisasi NMF dan

Extra Trees Classifier merupakan metode yang dapat digunakan untuk melakukan optimasi fitur.

1.2 Perumusan Masalah

Berdasarkan latar belakang masalah, maka perumusan masalah dalam penelitian ini adalah berapa akurasi penggunaan metode pemilihan fitur menggunakan faktorisasi NMF dan *Extra Trees Classifier* pada SVM dalam analisis sentimen terhadap data Twitter berdasarkan studi kasus calon presiden Republik Indonesia 2014?

1.3 Tujuan Penelitian

Sesuai dengan permasalahan yang diangkat, maka tujuan pada penulisan ini adalah studi komparasi akurasi penggunaan metode pemilihan fitur menggunakan faktorisasi NMF dan *Extra Trees Classifier* pada SVM dalam sentimen analisis terhadap data Twitter berdasarkan studi kasus calon presiden Republik Indonesia 2014.

1.4 Metodologi Penelitian

Untuk melakukan penelitian, dilakukan tahap-tahap sebagai berikut:

(a) Perumusan Masalah dan Studi Literatur

Pada tahap awal penelitian dilakukan perumusan masalah, tujuan, dan tahap-tahap untuk menyelesaikannya. Selain itu, studi literatur juga dilakukan untuk mendapatkan teori-teori yang mendukung baik melalui buku, jurnal, dan literatur lainnya.

(b) Akuisisi Data

Setelah tahap perumusan masalah dan studi literatur, dilakukan akuisisi data (pengumpulan data). Data tersebut kemudian akan dijadikan sebagai data sumber penelitian. Pada tahap ini, data yang dijadikan

sumber adalah data obrolan (*tweet*) dari para pengguna Twitter di Indonesia berkaitan dengan calon presiden.

(c) Implementasi Algoritma dan Simulasi

Pada tahap ini implementasi algoritma optimasi fitur dilakukan pada SVM melalui bahasa pemrogaman utama python.

(d) Interpretasi dan Analisa Hasil

Pada tahap terakhir penelitian, akan dilakukan analisa terkait kinerja dari penggunaan metode optimasi fitur pada SVM terhadap data sumber.

BAB 2
DASAR TEORI

2.1 Analisis Sentimen

Analisis sentimen merupakan suatu kegiatan untuk mendapatkan nilai sentimen dari suatu argumen yang diberikan oleh *author* berkaitan tentang entitas tertentu (Feldman, 2013). Ada dua metode yang umum digunakan untuk melakukan analisis sentimen, yaitu :

a.) *Lexicon-based Method*

Lexicon-based method adalah teknik analisis sentimen dengan menggunakan *lexicon* (kamus bahasa Inggris) yang tersedia secara umum untuk melakukan prediksi nilai sentimen, misalnya SentiWordnet.

b.) *Learning-based Method*

Learning-based method adalah teknik analisis sentimen yang menggunakan metode-metode *machine learning* untuk mendapatkan nilai sentimen, misalnya *Support Vector Machine* (SVM). *Learning-based method* akan memberikan prediksi nilai sentimen berdasar pola isi *tweet*, tidak tergantung bahasa dan juga dapat membaca *emoticon* dalam *tweet*.

2.2 *Machine learning*

Menurut Tom M. Mitchell (1997), definisi formal tentang *machine learning* sebagai berikut :

> "Sebuah program komputer dikatakan belajar dari pengalaman E yang bergantung pada target T dan ukuran kinerja program P jika kinerja pada target T, menggunakan ukuran P, ditingkatkan oleh pengalaman E."

Dengan kata lain, *machine learning* memungkinkan mesin untuk memberikan hasil prediksi pada data baru setelah mendapat pengalaman dari sekumpulan data (*data training*).

Berdasarkan input yang diberikan pada *data training*, *machine learning* dikelompokkan menjadi dua jenis, yaitu:

2.2.1 *Supervised Learning*

Pada *supervised learning*, *data training* disertai target pada setiap datanya, $\{\boldsymbol{x}_i, t_i\}, i = 1 \ldots N$, dengan $\boldsymbol{x}$ adalah vektor input dan t adalah target. Tujuan dari *supervised learning* adalah membangun model yang dapat memberikan hasil/output secara benar untuk suatu data input. *Supervised learning* digunakan untuk *classification*, *regression*, *ordinal regression*, *ranking*, dll.

2.2.2 *Unsupervised Learning*

Pada *unsupervised learning*, *data training* tidak disertai target pada setiap datanya, $\{\boldsymbol{x_i}\}, i = 1 \ldots N$, dengan $\boldsymbol{x}$ adalah vektor input. Tujuan dari *unsupervised learning* adalah membangun model yang dapat menemukan variabel tersembunyi dari *data training*. *Unsupervised learning* digunakan untuk *clustering, concept extraction*, *recommendation*, *density estimation*, *dimensionality reduction*, dll.

2.3 *Norm*

$\mathbb{R}^n$, menotasikan himpunan dari semua **vektor kolom** berdimensi n dengan koefisisen bilangan riil.

Definisi 2.3.1 (Burden, Fairies, & Julet, 2011, hal. 418)

Sebuah ***norm* vektor** pada $\mathbb{R}^n$ adalah sebuah fungsi, $\|.\|$, dari $\mathbb{R}^n$ ke $\mathbb{R}$ yang memenuhi sifat berikut:

1. $\|\boldsymbol{x}\| \geq 0$ untuk semua $\boldsymbol{x} \in \mathbb{R}^n$,
2. $\|\boldsymbol{x}\| = 0$ jika dan hanya jika $\boldsymbol{x} = 0$,
3. $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$ untuk semua $\alpha \in \mathbb{R}$ dan $\boldsymbol{x} \in \mathbb{R}^n$,

4. $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ untuk semua $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

Definisi 2.3.2 (Burden, Fairies, & Julet, 2011, hal. 418)

***Norm* Euclidean** dari vektor $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$ adalah

$$\|\boldsymbol{x}\| = \left\{\sum_{i=1}^{n} x_i^2\right\}^{1/2} . \tag{2.1}$$

Definisi 2.3.3 (Burden, Fairies, & Julet, 2011, hal. 421)

Jika $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$ dan $\boldsymbol{y} = (y_1, y_2, \dots, y_n)^T$ adalah vektor di $\mathbb{R}^n$, **jarak** antara $\boldsymbol{x}$ dan $\boldsymbol{y}$ didefinisikan dengan

$$\|\boldsymbol{x}-\boldsymbol{y}\| = \left\{\sum_{i=1}^{n} (x_i - y_i)^2\right\}^{1/2} . \tag{2.2}$$

2.4 Himpunan dan Fungsi Konveks / Konkaf

Definisi 2.4.1 (Bazaraa, Sherali, & Shetty, 2006, hal. 40)

Sebuah himpunan titik S di $\mathbb{R}^n$ dikatakan **himpunan konveks** jika garis yang menghubungkan kedua titik di dalam himpunan juga termasuk dalam himpunan S. Atau dengan kata lain, jika $\boldsymbol{x_1}$ dan $\boldsymbol{x_2}$ adalah anggota himpunan S, maka untuk setiap $\lambda \in [0,1]$,

$$\lambda \boldsymbol{x_1} + (1-\lambda)\boldsymbol{x_2}, \tag{2.3}$$

juga merupakan anggota himpunan S.

Definisi 2.4.2 (Winston, 2004, hal. 634)

Sebuah fungsi $f(x_1, x_2, \dots, x_n)$ adalah sebuah **fungsi konveks** pada **himpunan konveks** S jika untuk setiap $\boldsymbol{x}' \in S$ dan $\boldsymbol{x}'' \in S$,

$$f[c\boldsymbol{x}' + (1-c)\boldsymbol{x}''] \leq cf(\boldsymbol{x}') + (1-c)f(\boldsymbol{x}'') \tag{2.4}$$

terpenuhi untuk $0 \leq c \leq 1$.

Definisi 2.4.3 (Winston, 2004, hal. 634)

Sebuah fungsi $f(x_1, x_2, \dots, x_n)$ adalah sebuah **fungsi konkaf** pada **himpunan konveks** S jika untuk setiap $\boldsymbol{x}' \in S$ dan $\boldsymbol{x}'' \in S$,

$$f[c\boldsymbol{x}' + (1-c)\boldsymbol{x}''] \geq cf(\boldsymbol{x}') + (1-c)f(\boldsymbol{x}'') \tag{2.5}$$

terpenuhi untuk $0 \leq c \leq 1$.

Definisi 2.4.4 (Winston, 2004, hal. 637)

Matriks **Hessian** dari $f(x_1, x_2, \dots, x_n)$ adalah matriks $n \times n$ dengan elemen ke-ij adalah

$$\frac{\partial^2 f}{\partial x_i \partial x_j}. \tag{2.6}$$

Definisi 2.4.5 (Winston, 2004, hal. 637)

Principal minor ke-i dari matriks $n \times n$ adalah determinan dari setiap matriks $i \times i$ yang didapat dengan menghilangkan $n - i$ baris dan $n - i$ kolom dari matriks tersebut.

Teorema 2.4.6 (Winston, 2004, hal. 638)

Jika $f(x_1, x_2, \dots, x_n)$ memiliki turunan parsial kedua yang **kontinu** $\boldsymbol{x} = (x_1, x_2, \dots, x_n) \in S$, maka $f(x_1, x_2, \dots, x_n)$ adalah **fungsi konveks** jika dan hanya jika untuk setiap $\boldsymbol{x} \in S$, semua ***principal minor***-nya bernilai *nonnegatif* (**semidefinit positif**).

Teorema 2.4.7 (Winston, 2004, hal. 638)

Jika $f(x_1, x_2, \dots, x_n)$ memiliki turunan parsial kedua yang **kontinu** $\boldsymbol{x} = (x_1, x_2, \dots, x_n) \in S$, maka $f(x_1, x_2, \dots, x_n)$ adalah **fungsi konkaf** jika dan hanya jika untuk setiap $\boldsymbol{x} \in S$ dan $k = 1,2, \dots n$, semua ***principal minor*** yang bukan nol memiliki tanda seperti $(-1)^k$ (**semidefinit negatif**).

2.5 *Hyperplane*

Definisi 2.5.1 (Bazaraa, Sherali, & Shetty, 2006, hal. 52)

Sebuah ***hyperplane*** H pada $\mathbb{R}^n$ adalah himpunan titik dengan bentuk $\{\boldsymbol{x}: \boldsymbol{p^t x} = \alpha\}$, di mana $\boldsymbol{p}$ adalah vektor bukan nol di $\mathbb{R}^n$ dan α adalah skalar. Vektor $\boldsymbol{p}$ disebut vektor normal dari ***hyperplane***. Sebuah ***hyperplane*** H akan membentuk dua ***closed half-spaces*** $H^+ = \{\boldsymbol{x}: \boldsymbol{p^t x} \geq \alpha\}$ dan $H^- = \{\boldsymbol{x}: \boldsymbol{p^t x} \leq \alpha\}$, serta dua ***open half-spaces*** $\{\boldsymbol{x}: \boldsymbol{p^t x} > \alpha\}$ dan $\{\boldsymbol{x}: \boldsymbol{p^t x} < \alpha\}$.

Definisi 2.5.2 (Elizindo, 2006)

Dua himpunan X dan Y pada $\mathbb{R}^n$ dikatakan ***linearly separable*** jika ada sebuah ***hyperplane*** P pada $\mathbb{R}^n$ sedemikian sehingga $\forall \boldsymbol{x} \in X$ dan $\forall \boldsymbol{y} \in Y$ terletak pada sisi ***hyperplane*** yang berbeda.

2.6 Dualitas

Definisi 2.6.1 (Bazaraa, Sherali, & Shetty, 2006, hal. 259)

Jika diberikan masalah optimasi pada $\boldsymbol{x} \in \mathbb{R}^n$ dengan bentuk primal P sebagai berikut:

$Min\ f(\boldsymbol{x})$

dengan syarat

$$g_i(\boldsymbol{x}) \leq 0;\ \ i = 1, \dots, m \tag{2.7}$$

$$h_i(\boldsymbol{x}) = 0;\ \ i = 1, \dots, l \tag{2.8}$$

maka masalah **dual Lagrange** D adalah sebagai berikut:

$$\theta(\boldsymbol{u}, \boldsymbol{v}) := \inf\left\{f(\boldsymbol{x}) + \sum_{i=1}^{m} u_i g_i(x) + \sum_{i=1}^{l} v_i h_i(x)\right\}. \tag{2.9}$$

Vektor $\boldsymbol{u_i}$ dan $\boldsymbol{v_i}$ disebut **pengali Lagrange** atau **variabel dual** dan proses ini dinamakan **dualisasi**.

Teorema 2.6.2 (*Saddle Point*) (Bazaraa, Sherali, & Shetty, 2006, hal. 269)

Sebuah solusi $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ dengan $\boldsymbol{x} \in X$ dan $\boldsymbol{u} \geq 0$ adalah ***saddle point*** untuk **fungsi Lagrangian** $\phi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{x}) + \boldsymbol{u}^{\boldsymbol{T}} g(\boldsymbol{x}) + \boldsymbol{v}^{\boldsymbol{T}} h(\boldsymbol{x})$ jika dan hanya jika

$$\phi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = min\{\phi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{x} \in X\}, \tag{2.11}$$

$$g(\boldsymbol{x}) \leq 0, h(\boldsymbol{x}) = 0, \tag{2.12}$$

$$\mathbf{u}^{\mathbf{T}} g(\boldsymbol{x}) = 0. \tag{2.13}$$

Lebih lanjut lagi, $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ adalah ***saddle point*** jika dan hanya jika $\boldsymbol{x}$ dan $(\boldsymbol{u}, \boldsymbol{v})$ secara berurutan adalah solusi optimal dari masalah primal P dan masalah dual D tanpa ada jarak dualitas, atau berarti $f(\boldsymbol{x}) = \theta(\boldsymbol{u}, \boldsymbol{v})$.

2.7 Kondisi Karush-Kuhn-Tucker (KKT)

Jika diberikan masalah optimasi *non linear P* sebagai berikut

$$Max.(Min.) f(x_1, x_2, \dots, x_n)$$

dengan syarat

$$g_i(x_1, x_2, \dots, x_n) \leq b_i, \tag{2.14}$$

$$x_1, x_2, \dots, x_n \geq 0, \tag{2.15}$$

di mana $i = 1, \dots, m$.

Bila kendala atau syarat memiliki bentuk :

- $h(x_1, x_2, \dots, x_n) \geq b$, maka harus ditulis sebagai $-h(x_1, x_2, \dots, x_n) \leq -b$,
- $h(x_1, x_2, \dots, x_n) = b$, maka harus ditulis sebagai $h(x_1, x_2, \dots, x_n) \leq b$ dan $-h(x_1, x_2, \dots, x_n) \leq -b$.

Teorema 2.7.1 (Syarat Perlu Kondisi KKT) (Winston, 2004, hal. 678)

Jika masalah optimasi *P* merupakan masalah **peminimuman**. Jika $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ adalah solusi optimal dari masalah optimasi *P*, maka $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ harus memenuhi seluruh kendala pada masalah optimasi *P*, dan ada pengali $\lambda_1, \lambda_2, \dots, \lambda_m$ yang memenuhi:

$$\frac{\partial f(\mathbf{x})}{\partial x_j} + \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\mathbf{x})}{\partial x_j} \geq 0 \qquad (j = 1, 2, \dots, n) \tag{2.16}$$

$$\lambda_i [b_i - g_i(\boldsymbol{x})] = 0 \qquad (i = 1, 2, \dots, m) \tag{2.17}$$

$$\left(\frac{\partial f(\boldsymbol{x})}{\partial x_j} + \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \right) x_j = 0 \quad (j = 1, 2, \dots, n) \tag{2.18}$$

$$\lambda_i \geq 0 \qquad (i = 1, 2, \dots, m) \tag{2.19}$$

Teorema 2.7.2 (Syarat cukup kondisi KKT) (Winston, 2004, hal. 678)

Jika masalah optimasi *P* merupakan masalah **peminimuman**. Jika $f(x_1, x_2, \dots, x_n)$ dan $g_i(x_1, x_2, \dots, x_n)$ untuk $i = 1, 2, \dots, m$ adalah **fungsi konveks**, maka setiap titik $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ yang memenuhi hipotesis **Teorema 2.7.1** adalah solusi optimal untuk *P*.

2.8 Fungsi Kernel

Fungsi Kernel adalah suatu fungsi K yang mana untuk semua vektor input $\boldsymbol{x}$, $\boldsymbol{z}$ akan memenuhi kondisi

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle \tag{2.20}$$

di mana $\phi(.)$ adalah fungsi pemetaan dari ruang input ke ruang fitur (ruang dengan dimensi yang lebih tinggi). Atau dengan kata lain, fungsi kernel adalah fungsi hasil kali dalam *(inner product)* pada ruang fitur.

Misal diberikan $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2, \boldsymbol{x} = (x_1, x_2)$, dan $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. K merupakan fungsi kernel karena

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2$$

$$= \langle (x_1, x_2), (z_1, z_2) \rangle^2$$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= (x_1 z_1)^2 + 2(x_1 z_1)(x_2 z_2) + (x_1 z_1)^2$$

$$= \langle \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right), \left(z_1^2, \sqrt{2} z_1 z_2, z_2^2\right) \rangle$$

$$= \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle.$$

Terdapat beberapa fungsi kernel, yaitu :

- linear : $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i^T x_j}$,
- polinomial : $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\gamma \boldsymbol{x_i^T x_j} + r)^d, \ \gamma > 0$,
- Radial Basis Function (RBF) : $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\gamma \left\| \boldsymbol{x_i} - \boldsymbol{x_j} \right\|^2\right); \ \gamma > 0$,
- sigmoid : $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tanh(\gamma \boldsymbol{x_i^T x_j} + r)$,

(Hsu, Chang, & Lin, 2010).

Fungsi kernel digunakan untuk melakukan transformasi suatu data ke dalam dimensi ruang yang lebih tinggi dengan memberikan operasi kepada data pada ruang dimensi aslinya.

BAB 3
SUPPORT VECTOR MACHINE
(SVM)

3.1 Bentuk Dasar SVM

SVM merupakan salah satu metode *machine learning* untuk *pattern recognition*, yang dalam penelitian ini adalah *text recognition*. Algoritma SVM sendiri pertama kalinya dikembangkan oleh Vladimir Vapnik (Bishop, 2006, hal. 326). Pada dasarnya, SVM akan membuat suatu *hyperplane* atau suatu bidang yang merupakan fungsi klasifikator antara dua kelas dengan menggunakan konsep memaksimumkan *margin*.

Misal diberikan himpunan pasangan data sebanyak N sebagai berikut

$$\{\boldsymbol{x_i}, y_i\}, \qquad i = 1, \dots, N$$

dengan:

- $\boldsymbol{x_i} = [x_1, x_2, \dots, x_m]$ adalah vektor baris berdimensi m (banyaknya fitur),
- $y_i \in \{-1,1\}$ adalah nilai target (kelas) pada setiap vektor baris.

SVM akan membuat fungsi klasifikator dengan bentuk umum

$$y = sign(\boldsymbol{w^T x} + b) \tag{3.1}$$

dengan $\boldsymbol{w}$ adalah vektor yang merepresentasikan parameter bobot, dan b adalah bias atau error yang berupa skalar.

Hyperplane yang dihasilkan SVM akan mengelompokkan data menjadi dua kelas, yaitu kelas positif dan kelas negatif dengan ketentuan:

$\boldsymbol{w^T x_i} + b \geq 1$ untuk i dengan $y_i = +1$,

$\boldsymbol{w^T x_i} + b \leq -1$ untuk i dengan $y_i = -1$,

atau dapat ditulis sebagai berikut

$$y_i(\boldsymbol{w^T x_i} + b) \geq 1, \forall i. \quad (3.2)$$

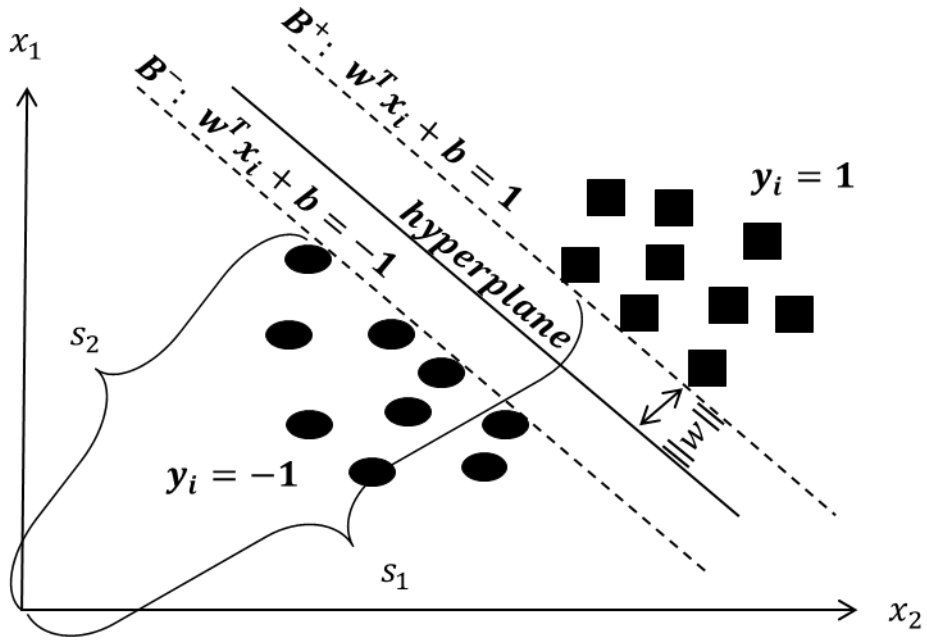
Misal dibentuk dua bidang yang saling sejajar dengan *hyperplane*, dengan satu bidang yang melewati titik terdekat data pada kelas positif (B^+),

$$B^+: \boldsymbol{w^T x_i} + b = 1 \quad (3.3)$$

dan satu bidang yang melewati titik terdekat data pada kelas negatif (B^-),

$$B^-: \boldsymbol{w^T x_i} + b = -1 \quad (3.4)$$

maka *margin* adalah jarak antara bidang B^+ dan B^-.



Gambar 3. 1 Ilustrasi *hyperplane* yang dihasilkan SVM

Untuk mendapatkan jarak terdekat (s_1) dari bidang B^+ terhadap pusat data dapat dilakukan dengan meminimumkan $\boldsymbol{x^T x}$ dengan syarat $\boldsymbol{w^T x_i} + b \geq 1$.

Dengan menggunakan pengali Lagrange (λ) dan turunan fungsinya terhadap $\boldsymbol{x_i}$ maka didapat formulasi sebagai berikut:

$$Min.\, \boldsymbol{x^T x} + \lambda(-\boldsymbol{w^T x} - b + 1) = 0 \tag{3.5}$$

$$\frac{\partial}{\partial \boldsymbol{x}} = 0 \rightarrow \boldsymbol{x^T} - \lambda \boldsymbol{w^T} = 0$$

$$\boldsymbol{x^T} = \lambda \boldsymbol{w^T} \rightarrow \boldsymbol{x} = \lambda \boldsymbol{w} \tag{3.6}$$

Dengan mensubstitusikan $\boldsymbol{x}$ pada persamaan (3.6) ke dalam persamaan bidang (3.3), B^+: $\boldsymbol{w^T x_i} + b = 1$, sehingga menjadi

$$\boldsymbol{w^T} \lambda \boldsymbol{w} + b = 1$$

$$\lambda = \frac{1 - b}{\boldsymbol{w^T w}} \tag{3.7}$$

Dengan mensubstitusikan λ pada persamaan (3.7) ke dalam $\boldsymbol{x}$ pada persamaan (3.6) menjadi

$$\boldsymbol{x} = \frac{1 - b}{\boldsymbol{w^T w}} \boldsymbol{w} = \frac{1 - b}{\boldsymbol{w^T}} \tag{3.8}$$

sehingga didapat persamaan untuk $\boldsymbol{x^T x}$ sebagai berikut:

$$\boldsymbol{x^T x} = \frac{(1 - b)^2}{\boldsymbol{w^T w}} \tag{3.9}$$

Jadi, jarak terdekat dari bidang B^+ ke bidang pemisah adalah

$$s_1 = \|\boldsymbol{x}\| = \sqrt{\boldsymbol{x^T x}} = \sqrt{\frac{(1 - b)^2}{\boldsymbol{w^T w}}} = \frac{(1 - b)}{\|\boldsymbol{w}\|} \tag{3.10}$$

Sedangkan untuk mendapatkan jarak terdekat (s_2) dari bidang B^- terhadap pusat data dapat dilakukan dengan meminimumkan $\boldsymbol{x^T x}$ dengan syarat $\boldsymbol{w^T x}$ +

$b \leq -1$. Dengan menggunakan pengali Lagrange (λ) dan turunan fungsinya terhadap x maka didapat formulasi sebagai berikut:

$$Min.\ \boldsymbol{x^T x} + \lambda(\boldsymbol{w^T x} + b + 1) = 0 \tag{3.11}$$

$$\frac{\partial}{\partial \boldsymbol{x}} = 0 \rightarrow \boldsymbol{x^T} + \lambda \boldsymbol{w^T} = 0$$

$$\boldsymbol{x^T} = -\lambda \boldsymbol{w^T} \rightarrow \boldsymbol{x} = -\lambda \boldsymbol{w} \tag{3.12}$$

Dengan mensubstitusikan $\boldsymbol{x}$ pada persamaan (3.12) ke persamaan bidang (3.4), B^-: $\boldsymbol{w^T x_i} + b = -1$, sehingga menjadi

$$-\boldsymbol{w^T} \lambda \boldsymbol{w} + b = -1$$

$$\lambda = \frac{1 + b}{\boldsymbol{w^T w}} \tag{3.13}$$

Dengan mensubstitusikan λ pada persamaan (3.13) ke dalam $\boldsymbol{x}$ pada persamaan (3.12) menjadi

$$x = -\frac{1 + b}{\boldsymbol{w^T w}} \boldsymbol{w} = \frac{-1 - b}{\boldsymbol{w^T}} \tag{3.14}$$

sehingga didapat persamaan untuk $\boldsymbol{x^T x}$ sebagai berikut:

$$\boldsymbol{x^T x} = \frac{(-1 - b)^2}{\boldsymbol{w^T w}} \tag{3.15}$$

Jadi, jarak terdekat dari bidang B^- ke *hyperplane* adalah

$$s_2 = \|\boldsymbol{x}\| = \sqrt{\boldsymbol{x^T x}} = \sqrt{\frac{(-1 - b)^2}{\boldsymbol{w^T w}}} = \frac{(-1 - b)}{\|\boldsymbol{w}\|} \tag{3.16}$$

Sehingga dari persamaan (3.10) dan (3.16) *margin* antara bidang B^+ dan bidang B^- adalah:

$$|s_1 - s_2| = \left|\frac{(1-b)}{\|\boldsymbol{w}\|} - \frac{(-1-b)}{\|\boldsymbol{w}\|}\right| = \left|\frac{2}{\|\boldsymbol{w}\|}\right| = \frac{2}{\|\boldsymbol{w}\|}. \quad (3.17)$$

Permasalahan pada SVM dapat diekspresikan menjadi berikut:

$$\max_{\boldsymbol{w},b}(margin) = \max_{\boldsymbol{w},b}\left(\frac{2}{\|\boldsymbol{w}\|}\right) \quad (3.18)$$

dengan syarat: $\min(\boldsymbol{w^T x_i} + b) = 1 \qquad i: y_i = +1,$

$\max(\boldsymbol{w^T x_i} + b) = -1 \qquad i: y_i = -1,$

sehingga akan menghasilkan fungsi klasifikator dengan bentuk sebagai berikut:

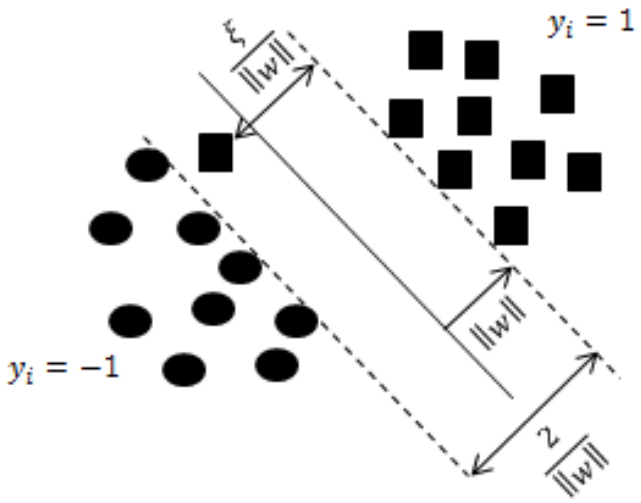
$$y = sign(\boldsymbol{w^T x} + b). \quad (3.19)$$

Masalah (3.18) juga dapat ditulis sebagai berikut:

$$\max_{\boldsymbol{w},b}\frac{1}{\|\boldsymbol{w}\|} \; atau \; \min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2 \quad (3.20)$$

dengan syarat:

$$y_i(\boldsymbol{w^T x_i} + b) \geq 1. \quad (3.21)$$



Gambar 3. 2 Ilustrasi timbulnya *error* dalam konstruksi SVM

Dalam pembentukan fungsi klasifikator, ada beberapa data yang tidak dapat diklasifikasikan secara benar (*misclassification error*) atau permasalahan optimasinya bersifat tidak layak (*infeasible*). Untuk menghindari masalah ini, ditambahkan variabel *slack* ($\xi_i \geq 0$) pada masalah optimasi (3.20) sehingga menjadi:

$$min\left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i\right) \tag{3.22}$$

dengan syarat :

$$y_i(\boldsymbol{w^T x_i} + b) \geq 1 - \xi_i, \tag{3.23}$$

$$\xi_i \geq 0 \quad , \forall i \tag{3.24}$$

di mana C akan mengontrol *tradeoff* antara variabel *slack* dan *margin*.

Permasalahan (3.22) merupakan bentuk *quadratic programming*. Untuk menyelesaikan pemrograman kuadrat tersebut, cara yang umum digunakan adalah mencari bentuk dual dengan menggunakan pengali Lagrange *(Lagrange multipliers*). Dari permasalahan di atas, bentuk *Lagrange*-nya adalah sebagai berikut:

$$\begin{aligned} \mathrm{L}(\mathbf{w}, \alpha, \lambda) &= \frac{1}{2}\|\mathbf{w}\|^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{N}}\xi_{\mathrm{i}} + \sum_{\mathrm{i}=1}^{\mathrm{N}}\alpha_{\mathrm{i}}(1 - \mathrm{y}_{\mathrm{i}}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + \mathrm{b}) - \xi_{\mathrm{i}}) \\ &\quad + \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}(-\xi_{\mathrm{i}}) \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{N}}\xi_{\mathrm{i}} + \sum_{\mathrm{i}=1}^{\mathrm{N}}\alpha_{\mathrm{i}} - \sum_{\mathrm{i}=1}^{\mathrm{N}}\alpha_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + \mathrm{b}) \\ &\quad - \sum_{\mathrm{i}=1}^{\mathrm{N}}\alpha_{\mathrm{i}}\xi_{\mathrm{i}} - \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\xi_{\mathrm{i}} \end{aligned} \tag{3.25}$$

dengan kendala variabel dualnya adalah sebagai berikut

$$\alpha_i \geq 0, \quad \lambda_i \geq 0. \tag{3.26}$$

Pada bentuk dual (3.25), α_i dan λ_i adalah pengali *Lagrange*.

Solusi dari masalah optimasi (3.25) dapat ditentukan melalui mencari *saddle point*-nya dengan cara menghitung turunan parsial dari L yang sama dengan 0 sebagai berikut:

- Turunan terhadap $\boldsymbol{w}$:

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} = 0$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} \tag{3.27}$$

- Turunan terhadap b :

$$\frac{\partial \mathrm{L}}{\partial \mathrm{b}} = \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.28}$$

- Turunan terhadap ξ_i :

$$\frac{\partial \mathrm{L}}{\partial \xi_i} = \mathrm{C} - \alpha_i - \lambda_i = 0 \tag{3.29}$$

dengan syarat $\lambda_i \geq 0, \quad \alpha_i \leq \mathrm{C}$.

Dengan mensubstitusikan turunan parsial L (3.27), (3.28), dan (3.29) ke masalah dual (3.25) menghasilkan:

$$\text{maks } L = \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}\right)\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}\right) + C\sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i$$
$$-\sum_{i=1}^{N} \alpha_i y_i \left(\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}\right) \cdot \mathbf{x_i} + b\right) - \sum_{i=1}^{N} \alpha_i \xi_i - \sum_{i=1}^{N} \lambda_i \xi_i$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j (\mathbf{x_i} \cdot \mathbf{x_j}) -$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j (\mathbf{x_i} \cdot \mathbf{x_j}) - b\sum_{i=1}^{N} \alpha_i y_i$$

$$+\sum_{i=1}^{N} \alpha_i + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \xi_i - \sum_{i=1}^{N} \lambda_i \xi_i$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j (\mathbf{x_i} \cdot \mathbf{x_j}) \quad (3.30)$$

dengan syarat

$$0 \leq \alpha_i \leq C, i = 1, \dots, N \quad (3.31)$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0. \quad (3.32)$$

Namun tidak semua data merupakan data yang dapat dipisahkan secara linier oleh *hyperplane* walaupun telah menambahkan variabel *slack*. Untuk mengatasi hal ini, digunakan fungsi kernel untuk memetakan data ke dimensi yang lebih tinggi sehingga diharapkan data menjadi bersifat *linearly separable*. Fungsi kernel *Radial Basis Function* (RBF) secara umum direkomendasikan sebagai pilihan utama, sehingga pada penelitian ini akan digunakan fungsi kernel RBF (Hsu, Chang, & Lin, 2010). Hal ini dikarenakan oleh 3 hal, yaitu:

- fungsi kernel RBF dapat memetakan secara tidak linear data ke ruang dimensi yang lebih tinggi sehingga diharapkan dapat menangani kasus di mana relasi antara kelas dan fitur tang tidak linear,
- fungsi kernel RBF menggunakan lebih sedikit parameter, dan
- fungsi kernel RBF memiliki kesulitan numerik yang lebih sedikit dibandingkan dengan fungsi kernel yang lain.

Dalam pemakaiannya, data $\mathbf{x_i}$ akan dipetakan dengan fungsi $\phi(\mathbf{x_i})$ dan setiap perkalian $(\mathbf{x_i} \cdot \mathbf{x_j})$ akan dihitung menggunakan $\mathrm{K}(\mathbf{x_i}, \mathbf{x_j})$ yang memiliki bentuk sebagai berikut

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\gamma \left\| \boldsymbol{x_i} - \boldsymbol{x_j} \right\|^2\right);\ \gamma > 0, \tag{3.33}$$

Dengan menambahkan penggunaan fungsi kernel RBF ke dalam persamaan (3.30), maka bentuk masalah dual yang baru adalah sebagai berikut :

$$\text{maks } \mathrm{L} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} - \frac{1}{2} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{y_i} \alpha_{\mathrm{j}} \mathrm{y_j} \mathrm{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{3.34}$$

dengan syarat

$$0 \leq \alpha_{\mathrm{i}} \leq \mathrm{C}, \mathrm{i} = 1, \ldots, \mathrm{N} \tag{3.35}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{y_i} = 0. \tag{3.36}$$

Permasalahan SVM dalam bentuk dual Lagrange (3.34) dapat diselesaikan dengan menggunakan metode *Sequential Minimal Optimization* (SMO) (Platt, 1998). Jika solusi dari permasalahan (3.34) adalah $\boldsymbol{\alpha}^*$, maka $\boldsymbol{w}$ dapat dicari dengan substitusi $\boldsymbol{\alpha}^*$ ke dalam persamaan (3.27) sebagai berikut:

$$\mathbf{w} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_i^* \mathrm{y_i} \mathbf{x_i}. \tag{3.37}$$

Dengan mensubstitusikan persamaan (3.37) ke dalam persamaan (3.1) sehingga fungsi klasifikator (*hyperplane*) adalah persamaan dengan bentuk sebagai berikut:

$$y = sign(\mathbf{w}^{\boldsymbol{T}}\boldsymbol{x} + b)$$

$$y = sign\left(\sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_i^* \mathrm{y_i} \mathbf{x}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{x} + b\right)$$

$$y = sign\left(\sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_i^* \mathrm{y_i} K(\boldsymbol{x}, \mathbf{x}_{\boldsymbol{i}}) + b\right). \tag{3.38}$$

Pada persamaan (3.38) hanya data dengan $\alpha_i^* > 0$ (*support vectors*) yang akan berperan pada model persamaan fungsi klasifikator (*hyperplane*) atau dengan kata lain

$$y = sign\left(\sum_{i \in S} \alpha_i^* \mathrm{y_i} K(\boldsymbol{x}, \mathbf{x}_{\boldsymbol{i}}) + b\right) \tag{3.39}$$

dengan S adalah himpunan indeks dari *support vectors*.

Bentuk dual dari persamaan (3.39) memenuhi kondisi KKT sebagai berikut:

$$\alpha_i \geq 0, \qquad [KKT-1]$$

$$\mathrm{y_i}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + \mathrm{b}) - 1 + \xi_\mathrm{i} \geq 0, \qquad [KKT-2]$$

$$\alpha_i(\mathrm{y_i}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + \mathrm{b}) - 1 + \xi_\mathrm{i}) = 0, \qquad [KKT-3]$$

$$\lambda_\mathrm{i} \geq 0, \qquad [KKT-4]$$

$$\xi_\mathrm{i} \geq 0, \qquad [KKT-5]$$

$$\lambda_\mathrm{i}\xi_\mathrm{i} = 0. \qquad [KKT-6]$$

Karena *support vectors* adalah *data training* dengan $\alpha_i^* > 0$, maka KKT – 3 menjadi

$$y_i(\mathbf{w^T x_i} + b) - 1 + \xi_i = 0$$

$$y_i(\mathbf{w^T x_i} + b) = 1 - \xi_i. \quad (3.40)$$

Dari syarat persamaan (3.35) dan dari hasil turunan *L* terhadap ξ_i pada persamaan (3.29), akan didapat:

- bila $\alpha_i^* < C$ maka $\lambda_i > 0$, dan berdasarkan KKT – 6 maka $\xi_i = 0$ yang berarti *data training* terletak pada *hyperplane*,
- bila $\alpha_i^* = C$ maka $\lambda_i = 0$, dan berdasarkan KKT – 6 maka $\xi_i \neq 0$ yang berarti *data training* terletak di dalam margin, baik terklasifikasi dengan benar ($\xi_i \leq 1$)maupun salah($\xi_i > 1$).

Dari fakta tersebut, maka *b* dapat dicari dengan persamaan sebagai berikut:

$$y_i(\mathbf{w^T x_i} + b) = 1$$

$$y_i\left(\sum_{j \in S} \alpha_j^* y_j K(\boldsymbol{x_i}, \mathbf{x_j}) + b\right) = 1$$

$$b = \frac{1}{N_S}\sum_{i \in S}\left(y_i \sum_{j \in S} \alpha_j^* y_j K(\boldsymbol{x_i}, \mathbf{x_j})\right). \quad (3.41)$$

dengan *S* adalah himpunan indeks dari *support vector*, dan N_S adalah banyaknya data train yang menjadi *support vectors*.

3.2 Konveks SVM

Dari persamaan (3.20) diperoleh bentuk dual Lagrange *L* sebagai berikut

$$\mathrm{L} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} - \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} \alpha_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\alpha_{\mathrm{j}}\mathrm{y}_{\mathrm{j}}\mathrm{K}(\mathbf{x_i}, \mathbf{x_j}).$$

Turunan pertama dari dual *L* adalah sebagai berikut

$$\frac{\partial L}{\partial \alpha_s} = 1 - \alpha_s K(\boldsymbol{x_s}, \mathbf{x_s}) - \sum_{i \neq s} \alpha_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\mathrm{y}_{\mathrm{s}}\mathrm{K}(\mathbf{x_i}, \mathbf{x_s}), \tag{3.42}$$

dengan $s = 1, \dots, N$

Turunan kedua dari dual *L* adalah sebagai berikut

$$\frac{\partial^2 L}{\partial \alpha_s \alpha_t} = -\boldsymbol{y_s y_t} K(\boldsymbol{x_s}, \boldsymbol{x_t}) \tag{3.43}$$

dengan $s = 1, \dots, N$ dan $t = 1, \dots, N$.

Matriks Hessian dari bentuk dual Lagrange *L* dapat ditulis sebagai

$$\nabla^2 L = -\boldsymbol{y}^T K_S \boldsymbol{y}, \tag{3.44}$$

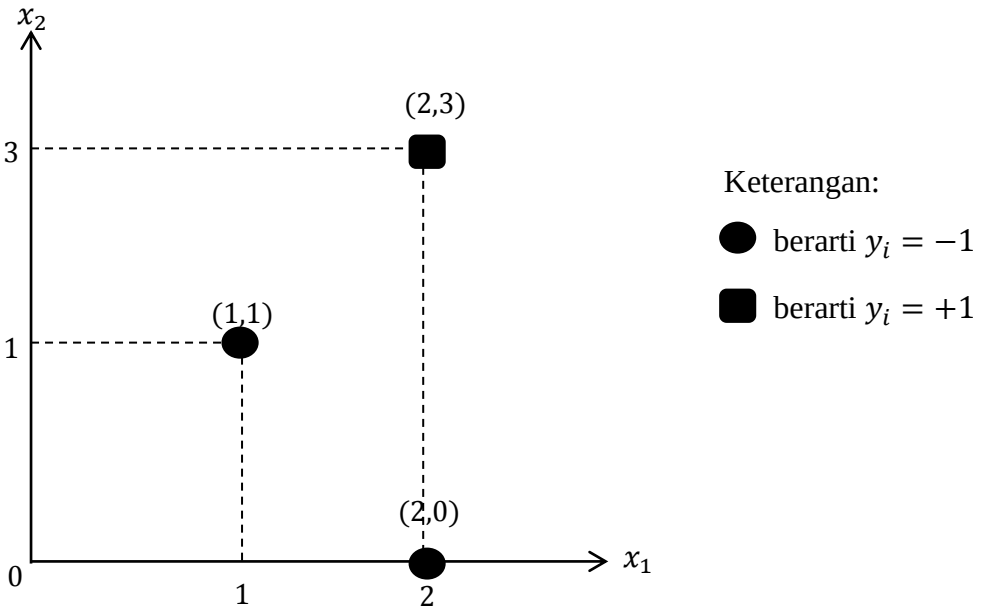
dengan K_s adalah matriks kernel untuk data sebanyak N. Fungsi kernel RBF memiliki sifat semidefinit positif (Cristianini & Shawe, 2000). Karena, fungsi kernel yang dipakai dalam penelitian ini adalah fungsi kernel RBF, maka $\boldsymbol{z}^T K_S \boldsymbol{z} \geq 0$ untuk semua vektor $\boldsymbol{z} \in \mathbb{R}^n$. Dengan kata lain,

$$\nabla^2 f = -\boldsymbol{y}^T K_S \boldsymbol{y} \leq 0, \tag{3.45}$$

sehingga bentuk dual Lagrange *L* bersifat konkaf. Karena bentuk dual Lagrange L dari permasalahan optimasi SVM bersifat konkaf, maka permasalahan primal dari SVM bersifat konveks. Hal ini berarti solusi dari SVM adalah solusi yang bersifat global optimum.

3.3 Contoh Penggunaan SVM

Misal diberikan *data training* $\left\{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}\right\} \in$ kelas (-1) dan $\begin{pmatrix}2\\3\end{pmatrix} \in$ kelas $(+1)$, atau data dapat digambarkan dengan fitur $\boldsymbol{x_1}$ dan $\boldsymbol{x_2}$ sebagai berikut



Gambar 3. 3 Contoh representasi geometrik dari data training dalam contoh

Perhatikan bahwa elemen terdekat dari masing-masing kelas adalah elemen $\begin{pmatrix}1\\1\end{pmatrix}$ dan $\begin{pmatrix}2\\3\end{pmatrix}$. Untuk membuat *hyperplane*, maka ambil satu titik yang pasti dilewati yaitu $\boldsymbol{w} = \begin{pmatrix}2\\3\end{pmatrix} - \begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}1\\2\end{pmatrix}$, yang akan mengimplikasikan vektor bobot $\boldsymbol{w} = \lambda\begin{pmatrix}1\\2\end{pmatrix} = \begin{pmatrix}\lambda\\2\lambda\end{pmatrix}$.

Ambil $\begin{pmatrix}1\\1\end{pmatrix} \in$ kelas(-1) maka dari persamaan (3.4) diperoleh

$$\boldsymbol{w^T x_i} + b = -1 \rightarrow (\lambda \quad 2\lambda)\begin{pmatrix}1\\1\end{pmatrix} + b = -1$$

$$\lambda + 2\lambda + b = -1$$

$$b = -1 - 3\lambda \tag{3.46}$$

Ambil $\begin{pmatrix}2\\3\end{pmatrix} \in$ kelas$(+1)$ maka dari persamaan (3.3) diperoleh

$$\boldsymbol{w^T x_i} + b = 1 \rightarrow (\lambda \quad 2\lambda)\begin{pmatrix}2\\3\end{pmatrix} + b = 1$$

$$2\lambda + 6\lambda + b = 1$$

$$b = 1 - 8\lambda \tag{3.47}$$

Dari persamaan (3.46) dan persamaan (3.47) didapatkan

$$-1 - 3\lambda = 1 - 8\lambda$$

$$5\lambda = 2$$

$$\lambda = \frac{2}{5}. \tag{3.48}$$

Dengan mensubstitusikan persamaan (3.48) ke dalam persamaan (3.46), maka diperoleh

$$b = -1 - 3\left(\frac{2}{5}\right) = -\frac{11}{5}. \tag{3.49}$$

Dari persamaan (3.48), maka vektor bobot *hyperplane* menjadi

$$\boldsymbol{w} = (\lambda, 2\lambda) = \left(\frac{2}{5}, \frac{4}{5}\right). \tag{3.50}$$

Sehingga *hyperplane* akan berbentuk sebagai berikut

$$y(\boldsymbol{x_1}, \boldsymbol{x_2}) = \boldsymbol{w^T x_i} + b$$

$$y(\boldsymbol{x_1}, \boldsymbol{x_2}) = \begin{pmatrix}\frac{2}{5} & \frac{4}{5}\end{pmatrix}\begin{pmatrix}\boldsymbol{x_1}\\\boldsymbol{x_2}\end{pmatrix} - \frac{11}{5}$$

$$y(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{2}{5}\boldsymbol{x_1} + \frac{4}{5}\boldsymbol{x_2} - \frac{11}{5}$$

Sehingga didapatkan *hyperplane* sebagai berikut:

$y(\boldsymbol{x_1}, \boldsymbol{x_2}) = 2\boldsymbol{x_1} + 4\boldsymbol{x_2} - 11.$

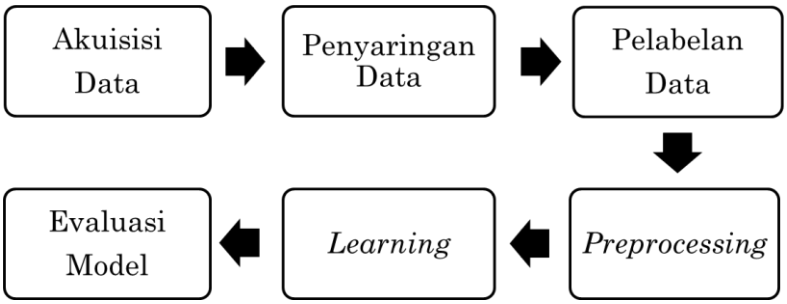
BAB 4
SIMULASI

4.1 Spesifikasi Mesin

Pada penelitian ini, simulasi dilakukan menggunakan program python dengan spesifikasi mesin sebagai berikut:

- Prosesor : Intel Atom
- Memori : 1 GB
- *Operating System* : Windows 7 (x32)
- Bahasa Pemrogaman : Python 2.7.3 [2]
- Modul Python : CSV, Numpy[3], Scipy[4], Pattern[5], sklearn[6].

4.2 Metodologi Penelitian



Gambar 4. 1 Skema Metode Penelitian

4.2.1 Akuisisi Data

Data teks yang digunakan pada penelitian ini berasal dari *tweet* para pengguna Twitter yang diambil melalui *software online* DiscoverText[7] dengan

[2] http://python.org
[3] http://numpy.org
[4] http://scipy.org
[5] http://www.clips.ua.ac.be/pattern
[6] http://scikit-learn.org
[7] http://discovertext.com

keyword "calon presiden". Akuisisi data dimulai pada bulan Agustus 2013 dan berakhir pada bulan Februari 2014 sehingga menghasilkan data sebanyak 11731 *tweet*.

4.2.2 Penyaringan Data

Data *tweet* dari para pengguna Twitter memiliki bermacam-macam tipe penulisan sehingga membuat mesin harus bekerja ekstra atau bahkan tidak bisa untuk memprosesnya. Untuk mengatasi hal ini, akan dilakukan penyaringan data sebelum data masuk ke dalam proses selanjutnya. Proses penyaringan data ini dilakukan melalui beberapa tahap, yaitu:

- Penghilangan karakter khusus

Mesin menggunakan pembacaan karakter dengan sistem *utf8-encoded*. Namun, tidak semua data *tweet* dapat dibaca oleh mesin dengan sistem *utf8-encoded* yang mengakibatkan data tidak dapat diproses. Untuk menanggulangi hal ini, akan dilakukan penghilangan terhadap beberapa karakter yang tidak didukung oleh mesin. Karakter-karakter yang akan dihilangkan tersebut diantaranya adalah karakter "±", "~", dan " …".

- Penghilangan kata khusus

Data *tweet* yang diambil pada penelitian ini masih banyak yang mengikutsertakan nama kandidat calon presiden. Hal ini akan memberi pengaruh pada pemberian nilai sentimen secara manual, terutama bila ada dua kandidat yang muncul pada satu data *tweet* yang sama. Selain itu, penggunakan *hashtag* (#) dan @ dalam *tweet* juga kurang memberi pengaruh pada pemberian nilai sentimen. Penggunaan *hashtag* kurang berpengaruh terhadap nilai sentimen karena penggunaan *hashtag* lebih fokus ke arah pembuatan topik. Sedangkan penggunaan @ dalam *tweet* hanya menunjukkan nama pengguna yang berinteraksi dalam suatu *tweet*. Untuk meningkatkan kinerja dari SVM, maka nama-nama kandidat presiden serta satu kata yang muncul setelah @ dan setelah

hashtag (#) akan dimasukkan ke dalam tabel '*stopwords*' yang kemudian tidak akan dipakai menjadi fitur pada tahap *preprocessing*.

- Penghilangan *tweet* yang mirip

Dalam Twitter, ada suatu pilihan yang bernama *retweet*. Teks hasil *retweet* sebagian besar atau bahkan seluruhnya merupakan *tweet* dari *user* lain walau terkadang ditambah dengan sedikit komentar dari *user* yang melakukan *retweet*. Hal ini mengakibatkan adanya data yang mirip dalam *data training*. Data yang mirip ini, tidak terlalu memberi pengaruh pada pembentukan fungsi klasifikator menggunakan SVM, namun hanya akan memperbesar dimensi data yang harus diolah oleh mesin. Algoritma *Levenshtein*[8] akan digunakan pada tahap ini untuk mengeliminasi data *tweet* yang mirip. Pada penelitian ini, dua data *tweet* akan diasumsikan mirip jika tidak memiliki perbedaan lebih dari lima belas karakter.

- Penghilangan *tweet* yang mengandung alamat website

Seiring dengan peningkatan aktivitas para *user* Twitter, banyak situs-situs *online*, terutama portal berita, yang menambahkan fitur tambahan untuk membagikan berita dengan tambahan alamat dari situs *online* berita tersebut ke dalam suatu *tweet*. *Tweet* ini biasanya mengandung kata '**http://**'. *Tweet* jenis ini tidak memiliki nilai sentimen karena hanya berdasar berita yang merupakan fakta.

Tabel 4. 1 Contoh *tweet* yang mengandung alamat website

Isi_*Tweet*
Menanti Jembatan Selat Sunda sebagai warisan SBY: Mantan presiden Megawati Soekarnoputri boleh berbangga denga... http://t.co/6czmJgLGqc
Janji yang tidak pernah ditepati oleh seorang mantan Presiden Megawati Soekarno Putri. Read more:... http://t.co/nAtW3KKJbx
Lika Liku Perjalanan Presiden Megawati Menjadi Presiden RI ke-5 - http://t.co/RmzEJbbQcx Simak langsung >>... http://t.co/NF9lAEaadR

[8] http://www.levenshtein.net/

Setelah dilakukan keempat tahap penyaringan data tersebut, maka didapatkan data bersih berjumlah 1189 *tweet* yang akan diproses lebih lanjut.

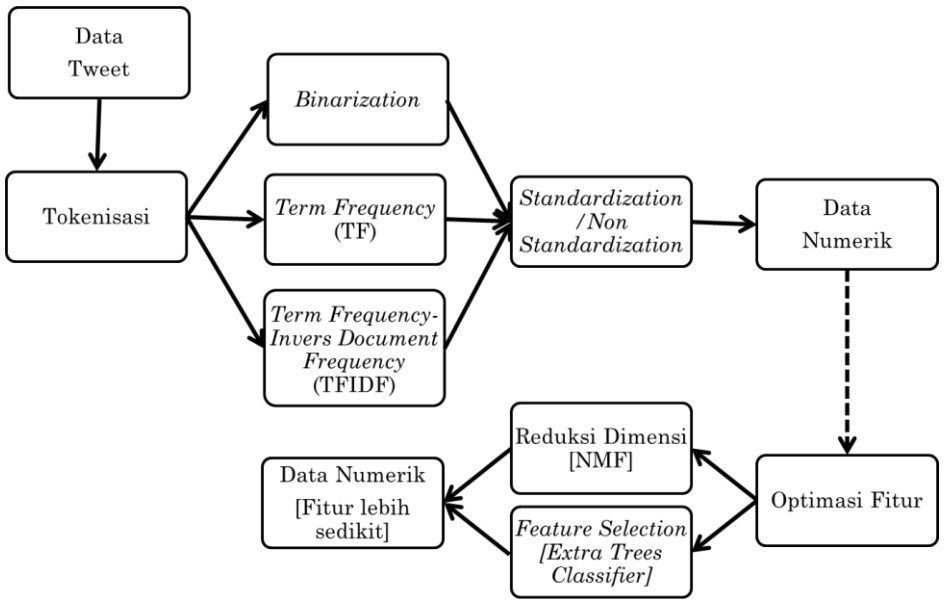
4.2.3 Pelabelan Data

Untuk memberikan pembelajaran pada mesin, dibutuhkan suatu *data training*. *Data training* ini dibuat dengan memberikan nilai sentimen secara manual pada setiap data *tweet* hasil dari proses penyaringan data. Pada penelitian ini, data *tweet* akan dikelompokkan menjadi data dengan nilai sentimen positif dan negatif. Data *tweet* yang bernilai sentimen positif akan diberi label dengan nilai '1', sedangkan data *tweet* yang bernilai sentimen negatif akan diberi label dengan nilai sentimen '-1'.

Tabel 4. 2 Contoh pelabelan *tweet* berdasar nilai sentimen

Isi_*Tweet*	**Sentimen**
Jangan plh capres arb dh ya. Bangun kampung halaman aja dya ga bsa, ngarep mau bangun Tanah Air	-1
eyaaa,coy jokowi ada dihati rakyat coy,pilpres rakyat mau jokowi jadi presiden!!,jokowi fenomenal coy !!!	1

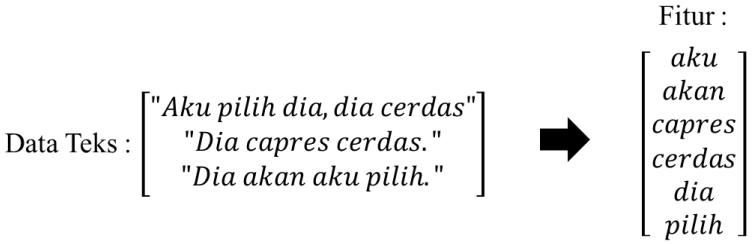
4.2.4 *Preprocessing*



Gambar 4. 2 Skema *preprocessing*

4.2.4.1 Tokenisasi

Tokenisasi adalah teknik untuk mengekstraksi fitur-fitur yang ada dalam keseluruhan data teks. Pada data teks, fitur adalah setiap kata unik yang ada dalam keseluruhan data teks. Jadi, pada tahap ini keseluruhan data teks akan dibaca kemudian dipartisi berdasarkan spasi dan tanda baca sehingga diperoleh fitur-fitur yang terdiri dari kata secara unik.



Gambar 4. 3 Contoh ekstraksi fitur pada tahap Tokenisasi

4.2.4.2 Metode Pembobotan

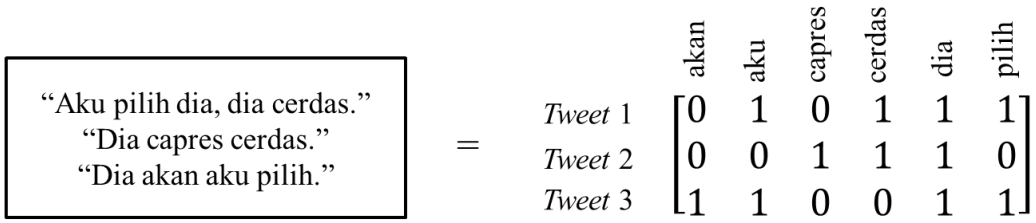
Pada tahap ini akan dilakukan proses ekstraksi data *tweet* yang tadinya masih berupa data teks menjadi vektor numerik yang dapat diproses oleh mesin. Pada dasarnya, proses ini akan melakukan ekstraksi fitur dari data teks yang berupa kata-kata yang saling berbeda. Kemudian data *tweet* akan direpresentasikan berdasarkan intensitas fitur-fitur yang muncul pada setiap data *tweet*. Dalam penelitian ini, akan dilakukan beberapa variasi dalam representasi data teks, yaitu

(1) *Binarization*

Binarization adalah salah satu teknik representasi data teks menjadi vektor numerik dengan elemen biner, yaitu 0 atau 1. Nilai 1 berarti munculnya fitur pada data teks, sedangkan nilai 0 merepresentasikan tidak munculnya suatu fitur pada data teks, atau $V = [v_{ij}], i = 1,2, \dots, N, j = 1,2, \dots, m$, dengan

$$v_{i,j} = \begin{cases} 0, & \text{bila tidak muncul fitur } j \text{ pada data teks } i \\ 1, & \text{bila muncul fitur } j \text{ pada data teks} \end{cases}$$

adalah vektor numerik hasil proses *binarization* (Manning, Raghavan, & Schütze, 2009).



Gambar 4. 4 Contoh vektor hasil dari proses *Binarization*

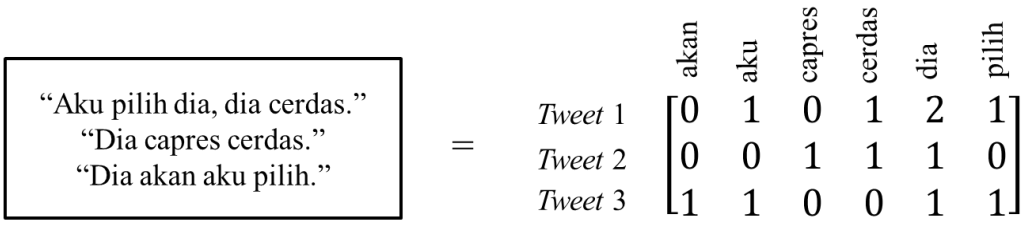
(2) *Term Frequency* (TF)

Metode ini merupakan proses umum untuk merepresentasikan data teks ke dalam vektor numerik dari fitur-fitur. Pada metode ini dilakukan penghitungan banyaknya fitur yang muncul pada data teks. Misalkan sebuah dokumen $\boldsymbol{d} = (t_1, t_2, \dots, t_m)$, di mana t_j merupakan kata ke-j dalam dokumen $\boldsymbol{d}$ dan m adalah banyaknya kata (fitur) yang berbeda dalam $\boldsymbol{d}$. Maka tahap TF akan melakukan pembobotan data *tweet* dengan persamaan sebagai berikut

$$tf_j(\boldsymbol{d}) = w_j, \tag{4.1}$$

di mana $tf_j(\boldsymbol{d})$ fungsi yang merepresentasikan intensitas munculnya kata t_j dalam dokumen $\boldsymbol{d}$, sedangkan w_j merepresentasikan banyaknya kemunculan kata t_j dalam dokumen $\boldsymbol{d}$ (Manning, Raghavan, & Schütze, 2009).

Pada tahap ini dihasilkan vektor numerik dengan kolomnya berupa fitur-fitur, sedangkan data teks akan direpresentasikan ke dalam vektor baris.



Gambar 4. 5 Contoh vektor hasil dari proses TF

(3) *Term Frequency Invers Document Frequency* (TFIDF)

Vektor numerik hasil pembobotan mengakibatkan beberapa kata yang kurang penting seperti “aku”, “dia”, “itu”, dan sebagainya memiliki representasi numerik yang besar karena kata-kata tersebut sangat sering muncul. Hal ini akan membuat tertutupnya beberapa fitur penting. Atau dengan kata lain munculnya fitur tidak penting dengan elemen data yang besar akan menutupi fitur penting dengan elemen data yang kecil. Jika hal ini terjadi, maka mesin akan membuat suatu fungsi klasifikator yang buruk. Untuk mengatasi hal ini, maka digunakan metode *Term Frequency Invers Document Frequency* (TFIDF).

Pada metode ini, proses pembobotan dilakukan dengan persamaan berikut:

$$\boldsymbol{tf} \times \boldsymbol{idf}, \tag{4.2}$$

dengan tf merupakan vektor hasil proses TF, sedangkan idf merupakan hasil dari persamaan berikut:

$$idf_j = \log_{10} \frac{N}{df_j}, \tag{4.3}$$

di mana idf_j merupakan nilai *inverse documen frequency* ($\boldsymbol{idf}$) dari fitur ke- j, sedangkan N merupakan banyaknya seluruh dokumen *tweet*, dan df_j merupakan banyaknya tweet dalam keseluruhan dokumen yang memuat kata j (Manning, Raghavan, & Schütze, 2009).

	akan	aku	capres	cerdas	dia	pilih
Tweet 1	0.000	0.176	0.000	0.176	0.000	0.176
Tweet 2	0.000	0.000	0.477	0.176	0.000	0.000
Tweet 3	0.477	0.176	0.000	0.000	0.000	0.176

Gambar 4. 6 Contoh vektor hasil dari proses TFIDF

4.2.4.3 Normalisasi

Pada tahap ini, vektor numerik akan dinormalkan melaui pembobotan ulang. Normalisasi dilakukan dengan memberikan bobot ulang setiap fitur sehingga setiap baris dalam vektor memiliki nilai *norm* satuan. Dengan kata lain, bila $V = [\boldsymbol{v_i}]$, $i = 1,2, \dots N$, dengan $\boldsymbol{v_i} = [v_1, v_2, \dots, v_m]$ merupakan vektor baris ke-i dari vektor V, maka $V' = [\boldsymbol{v_i'}]$ dengan

$$\boldsymbol{v_i'} = \frac{\boldsymbol{v_i}}{\|\boldsymbol{v_i}\|}, \qquad i = 1, \dots, m \tag{4.4}$$

di mana $\boldsymbol{v_i'} = [v_1', v_2', \dots, v_m']$ merupakan vektor baris ke-i dari vektor V', adalah vektor hasil normalisasi dari vektor V (Manning, Raghavan, & Schütze, 2009).

	akan	aku	capres	cerdas	dia	pilih
Tweet 1	0.000	0.378	0.000	0.378	0.755	0.378
Tweet 2	0.000	0.000	0.577	0.577	0.577	0.000
Tweet 3	0.500	0.500	0.000	0.000	0.500	0.500

Gambar 4. 7 Contoh vektor hasil dari proses Normalisasi dari vektor hasil TF

4.2.4.4 *Standardization*

Fungsi kernel RBF yang menggunakan asumsi bahwa semua fitur data terpusat di nol dan memiliki standar deviasi satuan. Jika sebuah fitur bernilai sangat besar dibandingkan fitur lainnya maka fitur ini akan mendominasi fungsi tujuan dan menutup kemungkinan estimator untuk mempelajari fitur-fitur kecil yang lain. Untuk mengatasi hal ini, harus dilakukan proses *standardization* pada data. *Standardization* dilakukan dengan memusatkan data sehingga data memiliki nilai *mean* nol dan memiliki nilai standar deviasi satuan. Dengan kata lain, bila diberikan $V = [v_{ij}]$ dengan $i = 1,2, \dots N$, $j = 1,2, \dots m$, v_j adalah vektor kolom ke-j dari matriks V, μ_j dan σ_j adalah *mean* dan standar deviasi dari vektor kolom v_j, maka $V'' = [v_{ij}'']$ dengan

$$v''_{ij} = \frac{v_{ij} - \mu_j}{\sigma_j}, \tag{4.5}$$

adalah vektor hasil *standardization* dari vektor V (Hogg, McKean, & Craig, 2012).

	akan	aku	capres	cerdas	dia	pilih
$Tweet\ 1$	−0.707	+0.707	−0.707	+0.707	+1.414	+0.707
$Tweet\ 2$	−0.707	−1.414	+1.414	+0.707	−0.707	−1.414
$Tweet\ 3$	+1.414	+0.707	−0.707	−1.414	−0.707	+0.707

Gambar 4. 8 Contoh vektor hasil dari proses *Standardization* dari vektor TF

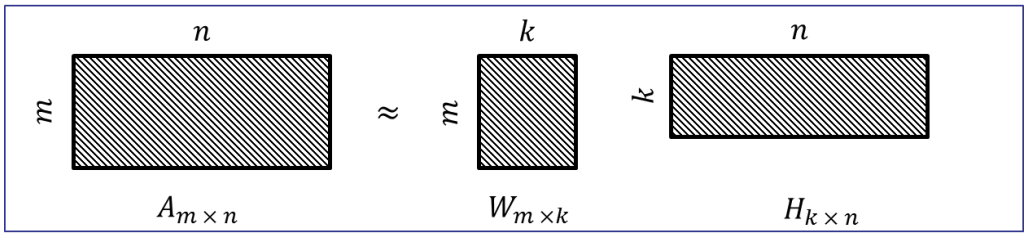
4.2.4.5 Metode Optimasi Fitur

(1) Reduksi Dimensi

Fitur hasil ekstraksi dari data teks berjumlah sangat banyak, sehingga akan mengakibatan vektor numerik dengan dimensi yang sangat besar. Mesin akan membutuhkan waktu yang lebih lama untuk memproses data ini. Untuk mempercepat kerja mesin, akan dilakukan dekomposisi dari vektor ini sehingga menghasilkan representasi data teks dengan dimensi fitur yang lebih kecil yang sering disebut reduksi dimensi. Dekomposisi vektor yang akan dipakai adalah *Nonnegative Matrix Factorization* (NMF).

Metode NMF merupakan faktorisasi untuk menguraikan suatu matriks $A_{m \times n}$ menjadi dua buah matriks, yaitu matriks $W_{m \times k}$ dan matriks $H_{k \times n}$ (Berry, 2006). Pada faktorisasi ini diasumsikan bahwa semua elemen matriks A adalah *nonnegative*. Pada faktorisasi ini, terdapat variabel:

- m, merupakan representasi banyaknya data,
- k, merupakan variabel tersembunyi yang merepresentasikan banyaknya topik, dan
- n, merupakan banyaknya kata hasil tokenisasi.



Gambar 4. 9 Skema faktorisasi NMF

(2) *Feature Selection*

Tidak semua fitur hasil ekstraksi data teks adalah fitur yang benar-benar berpengaruh dalam penentuan fungsi klasifikator. Banyak fitur bias yang tidak berpengaruh bahkan memperburuk kinerja mesin dalam menentukan fungsi klasifikator. Karena itu, akan dilakukan *feature selection* yang bertujuan untuk memilih fitur-fitur yang sangat berpengaruh terhadap penentuan fungsi klasifikator. Pada penelitian ini, *feature selection* yang digunakan berbasis *decision tree*, yaitu *Extra Trees Classifier* (ETC).

Tabel 4. 3 *Pseudo-code* dari algoritma *Extra Trees*

***Pseudo-code* dari algoritma *Extra Trees* (Geurts, Ernst, & Wehenkel, 2006)**
Build_an_extra_tree_ensemble(S). *Input*: himpunan *data training* S. *Output*: *ensemble tree* $T = \{t_1, \dots, t_M\}$. – Untuk $i = 1$ sampai *M* • Hasilkan sebuah *tree* : t_i =**Build_an_extra_tree**(S); – *Return* T. **Build_an_extra_tree**(S). *Input*: himpunan *data training* S. *Output*: sebuah *tree* t. – *Return* sebuah label *leaf* yaitu jumlah data dengan kelas terbanyak (atau rata-rata *output* untuk regresi) pada S jika: (i) $\lvert S \rvert < n_{min}$, atau (ii) Semua kandidat fitur bernilai konstan pada S, atau (iii) Variabel *output* pada S bernilai konstan. – Selain itu: 1. Pilih K fitur secara *random*, $\{a_1, \dots, a_K\}$, tanpa *replacement*, di antara (yang tidak bernilai konstan di S) kandidat fitur; 2. Hasilkan K *splits*, $\{s_1, \dots, s_K\}$, di mana s_i = **Pick_a_random_split**(S, a_i),

$\forall i = 1, \ldots, K$;
3. Pilih sebuah *split* s_* sedemikian sehingga Score$(s_*, S) = max_{i=1,\ldots,K}$ Score(s_i, S);
4. *Split S* menjadi subhimpunan S_l dan S_r berdasarkan s_*;
5. Buat $t_1 =$ **Build_an_extra_tree**$(\boldsymbol{S_l})$ dan $t_r =$ **Build_an_extra_tree**$(\boldsymbol{S_r})$ dari subhimpunan tersebut;
6. Buat sebuah *node* dengan *split* s_*, jadikan t_1 dan t_r sebagai *subtree* kanan dan kiri dari *node* tersebut dan *return tree* t yang dihasilkan.

Pick_a_random_split(S, a)
Input: himpunan *data training S* dan sebuah fitur a.
Output: sebuah *split*.
– Jika fitur a memiliki tipe data kontinu:
(i) Hitung nilai minimal dan maksimal dari a pada himpunan S,dinotasikan sebagai a^S_{min} dan a^S_{max};
(ii) Ambil sebuah *cut-point* a_c secara uniform dalam $[a^S_{min}, a^S_{max}]$;
(iii) *Return split* $[a < a_c]$.

– Jika fitur a memiliki tipe data kategorik (dinotasikan dengan A, yaitu himpunan semua nilai yang mungkin):
(i) Hitung A_S, subhimpunan A di mana nilai a muncul dalam S;
(ii) Ambil secara *random* sebuah subhimpunan tak kosong A_1 dari A_S dan sebuah subhimpunan $A_2 \in A \backslash A_S$;
(iii) *Return split* $[a \in A_1 \cup A_2]$.

di mana K adalah banyaknya fitur yang diambil secara *random* pada setiap *node*, dan n_{min} adalah banyaknya sampel minimum untuk melakukan pemisahan *node*. Score(s_*, S), dalam penelitian ini akan dihitung menggunakan persamaan sebagai berikut

$$\text{Score}(s_i, S) = \frac{2I^S_C(S)}{H_S(S) + H_C(S)}, \tag{4.6}$$

dengan

$$I^S_C(S) = H_C(S) - H_{C|s}(S), \tag{4.7}$$

$$H_S(S) = 1 - \sum_{i \in (l,r)} \left(\frac{|s_i|}{|S|}\right)^2, \tag{4.8}$$

$$H_C(S) = 1 - \sum_{i \in (1,-1)} p_i^2, \tag{4.9}$$

$$H_{C|s}(S) = 1 - \sum_{i\in(1,-1)} \sum_{j\in(\mathrm{l,r})} \frac{p^2(c_i, s_j)}{p(s_j)}, \tag{4.10}$$

di mana p menyatakan probabilitas.

4.2.6 *Learning*

Pada tahap ini, mesin akan diberikan *data training*, yaitu data kumpulan *tweet* yang disertai dengan pemberian nilai sentimen *per tweet* secara manual. *Data training* ini berfungsi memberikan mesin suatu pemahaman untuk dapat membedakan atau mengklasifikasikan *tweet* ke dalam *tweet* yang bernilai setuju (positif) atau tidak setuju (negatif). Selanjutnya, *data training* ini akan diproses menggunakan metode SVM dengan fungsi kernel *Radial Basis Function* (RBF) sehingga didapatkan model berupa fungsi klasifikator.

Untuk dapat menggunakan metode *training* SVM dengan fungsi kernel *Radial Basis Function* (RBF) dibutuhkan dua parameter, yaitu parameter C dan γ. Parameter C diperlukan oleh semua fungsi kernel SVM untuk mengatur *trade off* dari kesalahan pengklasifikasian *data training* terhadap penyederhanaan bidang. Nilai C yang kecil akan menyebabkan bidang tujuan halus dan nilai C yang besar bertujuan untuk dapat mengklasifikasikan semua *data training* secara benar. Sedangkan nilai γ akan merepresentasikan tingkat kepentingan sebagian *data training* terhadap *data training* lainnya. Nilai γ yang besar berarti sebagian *data training* akan semakin berpengaruh pada bagian lain dari *data training*.

Pemilihan kombinasi dari C dan γ akan mempengaruhi kinerja SVM. *Grid Search* digunakan untuk mendapatkan kombinasi yang optimal antara C dan γ (Hsu, Chang, & Lin, 2010). Proses untuk melakukan *Grid Search* secara lengkap memerlukan waktu yang sangat lama sehingga penelitian ini akan menggunakan metode pembagian *Grid Search* menjadi dua tahap, yaitu:

- *Loose Grid Search*

Dalam *Loose Grid Search*, dilakukan pemilihan parameter $C = 2^x$, dengan $x = -5, -3, \dots, 15$ serta $\gamma = 2^y$, dengan $y = -15, -13, \dots, 3$. Pemilihan

parameter ini dilakukan secara iteratif untuk setiap pemasangan C dan γ yang berbeda dan mengambil satu kombinasi $(C, \gamma) = (2^{C\_power}, 2^{\gamma\_power})$ di mana $C\_power = x$ dan $\gamma\_power = y$ yang menghasilkan kinerja terbaik untuk SVM.

- *Fine Grid Search*

Proses yang dilakukan pada tahap ini mirip seperti *Loose Grid Search*, namun kandidat C dan γ menjadi $C = 2^x$ dengan $x = C\_power - 2$, $C\_power - 1.75, \dots, C\_power + 2$ dan $\gamma = 2^y$ dengan $y = \gamma\_power - 2$, $\gamma\_power - 1.75, \dots, \gamma\_power + 2$.

Dalam penelitian ini, akan digunakan K-*Fold Cross Validation* untuk menghitung kinerja yang dihasilkan oleh SVM.

4.2.7 Evaluasi Model

Setelah diperoleh model fungsi klasifikator SVM yang terbaik, akan dilakukan evaluasi terhadap model tersebut. Hal ini dilakukan untuk mengetahui keakuratan model fungsi klasifikator dalam memprediksi data baru yang bukan termasuk dalam *data training*. K-*Fold Cross Validation* digunakan untuk menghitung akurasi model fungsi klasifikator terhadap data baru.

K-*Fold Cross Validation* pertama-tama dilakukan dengan membagi *data training* menjadi k bagian yang sama besar. Lalu $k - 1$ bagian tadi dijadikan *data training* baru dan bagian lain data dijadikan sebagai *data testing*. Proses ini dilakukan sebanyak k pengulangan pada setiap kombinasi *data testing* dan *data training* yang berbeda.

Untuk menyajikan hasil dari K-*Fold Cross Validation* ini digunakan *confusion matrix* dengan isi sebagai berikut:

		Target Prediksi	
		1	**-1**
Target Sebenarnya	**1**	True Positive (TP)	False Negative (FN)
	-1	False Positive (FP)	True Negative (TN)

Gambar 4. 10 Skema *Confusion Matrix*

Dari *confusion matrix*, menurut Tom Fawcet (2006) satuan kinerja dari model bisa dilihat dengan:

- *success rate* atau tingkat kesuksesan dapat dihitung dengan

$$\frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4.7}$$

- *True Positive Rate (TPR),* dapat dihitung dengan

$$\frac{TP}{(TP + FN)} \tag{4.8}$$

- *False Positive Rate (FPR),* dapat dihitung dengan

$$\frac{FP}{(FP + TN)} \tag{4.9}$$

4.3 Hasil Simulasi

4.3.1 Metode Representasi Data

Dalam penelitian ini akan dilakukan beberapa variasi representasi data. Variasi metode representasi data yang dilakukan pada penelitian ini adalah

1. TF – *Non Standardization* (NS),
2. TF – *Standardization* (S),
3. TFIDF – *Non Standardization* (NS),
4. TFIDF – *Standardization* (S),
5. *Binarization – Non Standardization* (NS),
6. *Binarization – Standardization* (S).

Dalam implementasi metode representasi data ini akan digunakan 5 – *Fold Cross Validation* untuk pengukuran kinerja pada setiap metode. Satuan kinerja yang digunakan dalam tahap ini adalah akurasi atau *succes rate* yang diinterpretasikan melalui tabel akurasi. Setelah dilakukan simulasi dengan menggunakan enam variasi metode pembobotan tersebut, menghasilkan akurasi sebagai berikut:

Tabel 4. 4 Perbandingan hasil akurasi metode representasi data

Metode Pembobotan	**Akurasi**	
	***Non Standardization* (NS)**	***Standardization* (S)**
TF	87 ± 0 %	66 ± 0 %
TFIDF	86 ± 1 %	86 ± 1 %
Binarization	87 ± 1 %	60 ± 0 %

Pada Tabel 4.4 terlihat bahwa:

- untuk metode representasi data TF, metode TF - NS memiliki akurasi lebih tinggi yaitu 87% daripada metode TF – S dengan akurasi 66%,
- untuk metode representasi data TFIDF, metode TFIDF – S maupun TFIDF – NS memiliki akurasi yang hampir sama yaitu (86 ± 1) %,
- untuk metode representasi data *Binarization*, metode *Binarization* – NS memiliki akurasi yang lebih tinggi yaitu (87 ± 1) % daripada metode *Binarization* – S dengan akurasi 60%.

Dari hasil simulasi pada tabel 4.4, proses representasi data TF – NS dan *Binarization* – NS akan dilanjutkan ke dalam tahap metode optimasi fitur karena memiliki tingkat akurasi yang lebih besar dibandingkan bila ditambah proses *standardization*. Sedangkan untuk metode representasi data TFIDF, metode TFIDF – S akan dilanjutkan ke tahap metode optimasi fitur, dikarenakan fungsi kernel yang dipakai dalam penelitian ini adalah fungsi kernel RBF.

4.3.2 Metode Optimasi Fitur

Ketiga metode representasi data yang telah terpilih pada proses sebelumnya memiliki akurasi yang cukup tinggi. Namun, vektor numerik hasil dari ketiga metode representasi data tersebut masih memiliki dimensi data yang sangat tinggi. Untuk mengurangi dimensi data vektor hasil, akan dilakukan optimasi fitur menggunakan metode faktorisasi NMF dan ETC pada setiap vektor hasil metode representasi data tersebut. Dalam metode optimasi fitur ini akan digunakan 5 – *Fold Cross Validation* dalam pengukuran kinerja pada setiap metode. Untuk pengukuran kinerja, satuan yang digunakan adalah akurasi atau *succes rate*, TPR, dan FPR yang akan diinterpretasikan melalui tabel akurasi dan grafik ROC. Pada grafik ROC, metode yang memiliki luas area area lebih besar akan memiliki kinerja yang lebih bagus (Fawcett, 2006).

4.3.2.1 Metode Optimasi Fitur berdasar TF – NS

Pada tahap ini, akan dilakukan metode optimasi fitur berdasar vektor hasil metode representasi data TF – NS. Setelah dilakukan metode optimasi fitur, hasil yang didapatkan adalah sebagai berikut:

- Tabel akurasi

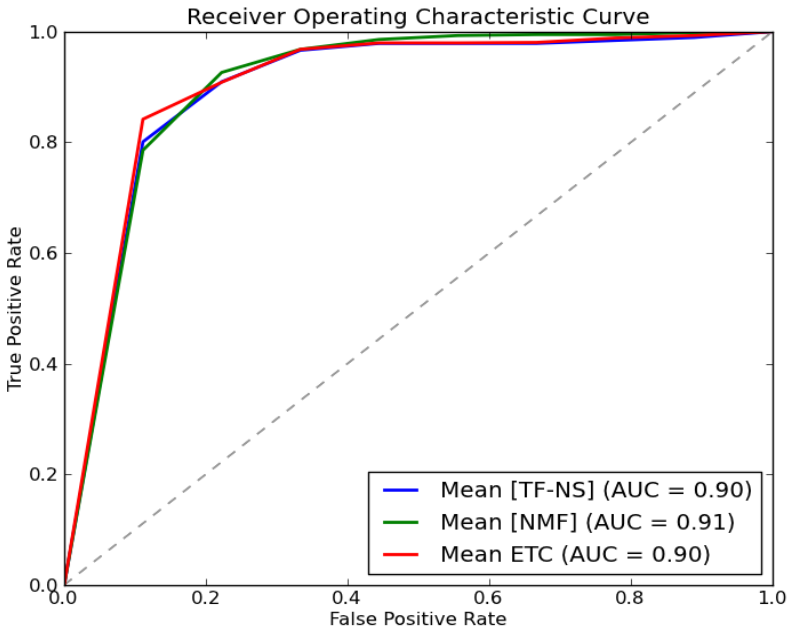
Tabel 4.5 Perbandingan hasil akurasi metode optimasi fitur berdasar TF – NS

Metode	Jenis Fitur	Banyak Data	Banyak Fitur	Rata-rata Akurasi
TF – NS	Kata	1889	2150	87 ± 0 %
ETC	Kata	1889	552	89 ± 1 %
NMF	Topik	1889	250	87 ± 0 %

Dari Tabel 4.5, terlihat bahwa penggunaan metode ETC mengurangi jumlah fitur sehingga dimensi fitur data hanya 552 fitur. Selain itu, ETC juga

meningkatkan akurasi data sehingga mencapai (89 ± 1) %. Sedangkan untuk metode faktorisasi NMF, terlihat bahwa metode ini dapat mengurangi dimensi fitur data menjadi 250 buah fitur namun dengan tetap mempertahankan akurasi metode TF – NS, yaitu 87 %.

- Grafik ROC



Gambar 4. 11 Grafik perbandingan ROC metode optimasi fitur berdasar metode TF – NS

Dari Gambar 4.11, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode faktorisasi NMF memiliki luas yang paling besar. Hal ini berarti bahwa metode faktorisasi NMF memiliki kemampuan prediksi yang paling tinggi dibandingkan dengan metode ETC maupun metode TF – NS.

Untuk metode ETC, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode ini memiliki luas yang sama seperti metode TF – NS . Hal ini berarti bahwa metode ETC memiliki kemampuan yang hampir sama dengan metode TF – NS. Namun, perlu diperhatikan pada Tabel 4.5 bahwa dimensi fitur data yang

dihasilkan setelah dikenai metode ETC jauh lebih sedikit dibandingkan metode TF – NS.

4.3.2.2 Metode Optimasi Fitur berdasar TFIDF – S

Pada tahap ini, akan dilakukan metode optimasi fitur berdasar vektor hasil metode representasi data TFIDF – S. Setelah dilakukan metode optimasi fitur, hasil yang didapatkan adalah sebagai berikut:

- Tabel akurasi

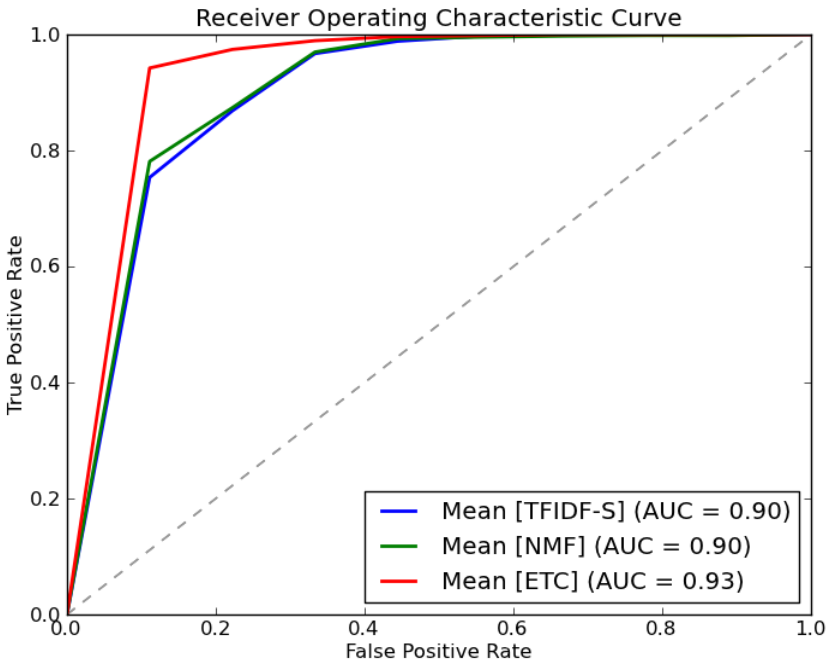
Dari Tabel 4.6, terlihat bahwa penggunaan metode ETC mengurangi jumlah fitur sehingga dimensi fitur data hanya 552 fitur. Selain itu, ETC juga meningkatkan akurasi data sehingga mencapai (92 ± 1) %. Sedangkan untuk metode faktorisasi NMF, terlihat bahwa metode ini dapat mengurangi dimensi fitur data menjadi 250 buah fitur namun dengan tetap mempertahankan akurasi metode TFIDF – S, yaitu 87 %.

Tabel 4. 6 Perbandingan hasil akurasi metode optimasi fitur berdasar metode TFIDF – S

Metode	Jenis Fitur	Banyak Data	Banyak Fitur	Rata-rata Akurasi
TFIDF – S	Kata	1889	2150	87 ± 0 %
ETC	Kata	1889	552	92 ± 1 %
NMF	Topik	1889	250	87 ± 0 %

- Grafik ROC

Dari Gambar 4.12, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode ETC memiliki luas yang paling besar. Hal ini berarti bahwa metode ETC memiliki kemampuan prediksi yang paling tinggi dibandingkan dengan metode faktorisasi NMF maupun metode TFIDF – S.



Gambar 4. 12 Grafik perbandingan ROC metode optimasi fitur berdasar metode TFIDF – S

Untuk metode faktorisasi NMF, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode ini memiliki luas yang sama seperti metode TFIDF – S . Hal ini berarti bahwa metode faktorisasi NMF memiliki kemampuan yang hampir sama dengan metode TFIDF – S . Namun, perlu diperhatikan pada Tabel 4.6 bahwa dimensi fitur data yang dihasilkan setelah dikenai metode faktorisasi NMF jauh lebih sedikit dibandingkan metode TFIDF – S.

4.3.2.3 Metode Optimasi Fitur berdasar *Binarization* – NS

Pada tahap ini, akan dilakukan metode optimasi fitur berdasar vektor hasil metode representasi data *Binarization* – NS. Setelah dilakukan metode optimasi fitur, hasil yang didapatkan adalah sebagai berikut:

- Tabel akurasi

Dari Tabel 4.7, terlihat bahwa penggunaan metode ETC mengurangi jumlah fitur sehingga dimensi fitur data hanya 552 fitur. Selain itu, ETC juga meningkatkan akurasi data sehingga mencapai 88 %. Sedangkan untuk metode

faktorisasi NMF, terlihat bahwa akurasi dari metode ini justru berkurang sehingga menjadi (86 ± 1) %. Namun perlu diingat bahwa dimensi fitur data setelah menggunakan metode faktorisasi NMF menjadi 250 buah fitur, jauh berkurang dari data hasil metode *Binarization* – NS.

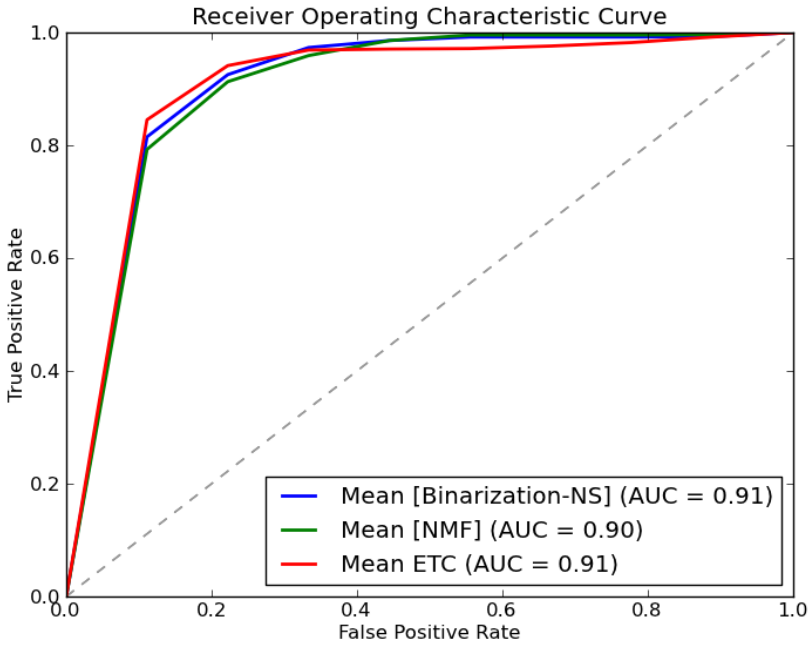
Tabel 4. 7 Perbandingan hasil akurasi metode optimasi fitur berdasar metode *Binarization* – NS

Metode	Jenis Fitur	Banyak Data	Banyak Fitur	Rata-rata Akurasi
Binarization – NS	Kata	1889	2150	87 ± 0 %
ETC	Kata	1889	552	88 ± 0 %
NMF	Topik	1889	250	86 ± 1 %

- Grafik ROC

Dari Gambar 4.13, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode ETC memiliki luas yang sama besar dengan metode *Binarization* – NS. Hal ini berarti bahwa metode ETC dan metode *Binarization* – NS memiliki kemampuan prediksi yang hampir sama. Namun, perlu diperhatikan pada Tabel 4.7 bahwa dimensi fitur data yang dihasilkan setelah dikenai metode ETC jauh lebih sedikit dibandingkan metode *Binarization* – NS.

Untuk metode faktorisasi NMF, terlihat bahwa luas daerah di bawah kurva (AUC) dari metode ini memiliki luas yang paling kecil dibandingkan metode ETC dan metode *Binarization* – NS. Hal ini berarti bahwa metode NMF memiliki kinerja di bawah metode *Binarization* – NS dan metode ETC. Namun, perlu diperhatikan pada Tabel 4.7 bahwa dimensi fitur data yang dihasilkan setelah dikenai metode faktorisasi NMF jauh lebih sedikit dibandingkan metode *Binarization* – NS maupun setelah dikenai metode ETC.



Gambar 4. 13 Grafik perbandingan ROC metode optimasi fitur berdasar metode *Binarization* – NS

BAB 5
KESIMPULAN DAN SARAN

Setelah melakukan tinjauan pustaka, akuisisi data, sampai implementasi SVM untuk melihat pengaruh optimasi fitur terhadap akurasi model pada masalah analisis sentimen terhadap data Twitter berdasarkan studi kasus calon presiden Republik Indonesia 2014, maka dapat diambil kesimpulan dan saran untuk pengembangan penelitian yang lebih lanjut sebagai berikut:

5.1 Kesimpulan

Dari percobaan implementasi simulasi untuk melihat pengaruh optimasi fitur menggunakan metode SVM terhadap akurasi analisis sentimen terhadap data Twitter berdasarkan studi kasus calon presiden Republik Indonesia 2014, maka didapatkan kesimpulan berikut:

1. Untuk metode dasar representasi data, metode TF – *Non Standardization* (NS) dan Binarization – *Non Standardization* (NS) memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan metode pembobotan TF dan *Binarization* yang diberikan prosedur tambahan *Standardization* (S);

2. Untuk metode optimasi fitur berdasar metode pembobotan TF – NS, penggunaan metode NMF mempertahankan akurasi metode TF – NS, sedangkan metode *Extra Trees Classifier* meningkatkan akurasi metode TF – NS;

3. Untuk metode optimasi fitur berdasar metode pembobotan TFIDF – S, penggunaan metode NMF mempertahankan akurasi metode TFIDF – S, sedangkan metode *Extra Trees Classifier* meningkatkan akurasi metode TFIDF – S;

4. Untuk metode optimasi fitur berdasar metode pembobotan *Binarization* – NS, penggunaan metode NMF mempertahankan akurasi metode *Binarization* – NS,

sedangkan metode *Extra Trees Classifier* meningkatkan akurasi metode *Binarition* – NS.

5. Dari seluruh metode yang telah diimplementasikan dalam penelitian ini, metode pembobotan TFIDF dengan penambahan proses *Standardization* dan dilanjutkan optimasi fitur *Extra Trees Classifier* memberikan akurasi terbaik dibandingkan dengan metode-metode yang lain.

5.2 Saran

Setelah melakukan percobaan implementasi melaui penelitian ini, masih diperlukan perbaikan untuk penelitian selanjutnya, yaitu:

1. diperlukannya algoritma untuk membaca kata-kata yang tidak baku ataupun kata-kata yang merupakan singkatan. Banyak kata-kata singkatan maupun kata-kata tidak baku lain yang memiliki arti sama dengan kata-kata baku lain dalam *tweet* yang berbeda. Hal ini menimbulkan tingginya dimensi fitur dalam *data training*;

2. diperlukannya *software* yang gratis dan *open source* untuk akuisisi data sehingga penelitian dapat terus berlanjut tanpa terkendala oleh ketidaksediaan data;

3. pemilihan parameter C dan γ terbaik melaui *Grid Search* dapat dilakukan secara paralel. Hal ini dikarenakan dalam menggunakan metode *Grid Search*, pencarian kombinasi C dan γ bersifat saling tak bergantung.

DAFTAR PUSTAKA

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). *Nonlinear Programming Theory and Algorithms.* USA: John Wiley & Sons, Inc.

Berry, M. W. (2006). Algorithms and Applications for Approximate Nonnegative Matrix Factorization. *Computational Statistics & Data Analysis*, *52*, 155 - 173.

Bishop, C. H. (2006). *Pattern Recognition and Machine Learning.* Springer.

Burden, R. L., Fairies, J. D., & Julet, M. (2011). *Numerical Analysis* (9th ed.). USA, Boston: Brooks/Cole Publishing Co.

Cristianini, N., & Shawe, T. J. (2000). *An Introduction to Support Vector Machines.* UK: Cambridge University Press.

Elizindo, D. (2006). The Linear Separability Problem: Some Testing Methods. In *IEEE Transactions on Neural Networks* (Vol. 17).

Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, *27*, 861-874.

Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. In *Communication of the ACM* (Vol. 56, pp. 82-89).

Geurts, P., Ernst, D., & Wehenkel, L. (2006, March 2). Extremely Randomized Trees. Springer Science+Business Media, Inc.

Hogg, R. V., McKean, J., & Craig, A. T. (2012). *Introduction to Mathematical Statistics* (7th ed.). Pearson.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2010, April 15). *A Practical Guide to Support Vector Classification.* Retrieved from http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Mitchell, T. M. (1997). *Machine Learning.* USA: McGraw Hill.

Platt, J. C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* Microsoft Research. Microsoft Research.

Winston, W. (2004). *Operation Research Applications and Algorithms* (4th ed.). USA: Brooks/Cole Publishing Co.

LAMPIRAN

Lampiran 1

Source code untuk menyaring data menggunakan *Levenshtein.*

```
import csv
import numpy as np

def levenshtein(a,b):
    "Hitung jarak Levenshtein string a dan b."
    n, m = len(a), len(b)
    if n > m:
        # Make sure n <= m, to use O(min(n,m)) space
        a,b = b,a
        n,m = m,n

    current = range(n+1)
    for i in range(1,m+1):
        previous, current = current, [i]+[0]*n
        for j in range(1,n+1):
            add, delete = previous[j]+1, current[j-
1]+1
            change       = previous[j-1]
            if a[j-1] != b[i-1]:
                change  = change + 1
            current[j]  = min(add, delete, change)

    return current[n]

def ambilTweet(targetTweet,kumpulanTweet):
     for tweet in kumpulanTweet:
          if levenshtein(tweet,targetTweet) <= 15:
               return 0
     return 1

table     = Datasheet.load('Tweet_Capres.csv')
try:
     table1    = Datasheet.load('data_reinforce.csv')
except:
     table1    = Datasheet()
     table1.append(['ID', 'Tweet_ID', 'Isi_Tweet',
'Tanggal_Tweet', 'Kandidat', 'Aplikasi', 'Lokasi
User','Sentimen'])

with open('Tweet_Capres.csv', 'rb') as data:
     reader         = csv.reader(data)
     y              = []
     tweet          = []
```

(Lanjutan)

```
     targetpositif  = 0
     targetnegatif  = 0

     for row in reader:
          if ambilTweet(row[2], tweet) == 1:
               if row[7] == '-1':
                    table1.append([row[0], row[1],
row[2], row[3], row[4], row[5], row[6], row[7]])
                    y    = np.append(y,[-1])
                    tweet     =
np.append(tweet,[row[2]])
                    targetnegatif  = targetnegatif+1
               if row[7] == '1':
                    table1.append([row[0], row[1],
row[2], row[3], row[4], row[5], row[6], row[7]])
                    y    = np.append(y, [1])
                    tweet     = np.append(tweet,
[row[2]])
                    targetpositif  = targetpositif +
1
     table1.save('data_reinforce.csv')
```

Lampiran 2

Source code perbandingan metode representasi data

```
import csv
import numpy as np
import scipy as sp
from pattern.db import Datasheet
from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer
from sklearn import preprocessing, cross_validation,
svm
from sklearn.cross_validation import StratifiedKFold
from sklearn.svm import SVC
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import time

def getStopwords(namaTabelStopwords):
     with open(namaTabelStopwords, 'rb') as data:
          reader     = csv.reader(data)
          stopword = []
          for row in reader:
               a              = row[0]
               stopword = stopword + [a]
          return stopword

def preprocessingTweet(tabelTweet, tabelStopword):
     metode = ['[TF-NS]', '[TF-S]', '[TFIDF-NS]',
'[TFIDF-S]', '[Binarization-NS]', '[Binarization-S]']

     #TF-NS
     CV   = CountVectorizer(stop_words =
tabelStopword).fit(tabelTweet)
     data = CV.transform(tabelTweet)
     X1   = data.toarray()

     #TF-S
     scaler    =
preprocessing.StandardScaler().fit(X1)
     X2   = scaler.transform(X1)

     #pakai TFIDF-NS
     TV   = TfidfVectorizer(stop_words =
tabelStopword)
     data = TV.fit_transform(tabelTweet)
     X3   = data.toarray()

     #pakai TFIDF-S
     scaler    =
preprocessing.StandardScaler().fit(X3)
```

(Lanjutan)

```
     X4   = scaler.transform(X3)

     #Binarization-NS (0 atau 1)
     #data > 0.9 menjadi bernilai 1, selain itu
     nilainya 0
     biner     =
preprocessing.Binarizer(threshold=0.9).fit(X1)
     X5   = biner.transform(X1)

     #Binarization-S
     scaler     =
preprocessing.StandardScaler().fit(X5)
     X6    = scaler.transform(X5)

     print "Dimensi Data Train TF-NS: ",X1.shape
     print "Dimensi Data Train TF-S:",X2.shape
     print "Dimensi Data Train TFIDF-NS:", X3.shape
     print "Dimensi Data Train TFIDF-S:", X4.shape
     print "Dimensi Data Train Binarization-NS:",
X5.shape
     print "Dimensi Data Train Binarization-S:",
X6.shape

     dataTrain =    [X1, X2, X3, X4, X5, X6]

     return [metode, dataTrain]

def trainData(arrayInput, arrayNilaiSentimen,
metode):
     tic = time.clock()

     ###Loose Grid-Search
     C_power    = np.arange(-5, 17, 2)
     gamma_power = np.arange(-15, 5, 2)
     C_range    = 2.0**C_power
     gamma_range  = 2.0**gamma_power
     param_grid      = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a         = grid.fit(arrayInput,
arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_gamma     =
np.log2(grid.best_params_.values()[1])

     print "best C : ",best_C, "best gamma : ",
best_gamma
```

(Lanjutan)

```
     # Extracting Scores
     score_dict = grid.grid_scores_
     scores = [x[1] for x in score_dict]
     scores = np.array(scores).reshape(len(C_range),
len(gamma_range))

     ###Fine Grid-Search
     C_power   = np.arange(best_C-2, best_C+2.25,
0.25)
     gamma_power = np.arange(float(best_gamma)-2,
best_gamma+2.25, 0.25)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a         = grid.fit(arrayInput,
arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_gamma  =
np.log2(grid.best_params_.values()[1])

     print "best C : ",best_C, "best gamma : ",
best_gamma
     # Extracting Scores
     score_dict = grid.grid_scores_
     scores = [x[1] for x in score_dict]
     scores = np.array(scores).reshape(len(C_range),
len(gamma_range))

     toc = time.clock()
     waktu = toc-tic

     print "Waktu komputasi untuk metode", metode,
"adalah ", waktu, "detik.\n"
     return [a, best_gamma, best_C]


#load tweet bahasa indonesia
with open('data_reinforce.csv', 'rb') as data:
     reader         = csv.reader(data)
     y         = []
     tweet          = []

     for row in reader:
          if row[7] == '-1':
               y    = np.append(y,[-1.0])
               tweet     = np.append(tweet,[row[2]])

          if row[7] == '1':
```

(Lanjutan)

```
                y    = np.append(y, [1.0])
                tweet     = np.append(tweet, [row[2]])



stopword         = getStopwords('stopwords.csv')
 [metode, dataTrain]       =
preprocessingTweet(tweet, stopword)
best_gamma            = np.zeros(len(metode))
best_C                = np.zeros(len(metode))

#compute Loose Grid and Fine Grid
for j in np.arange(len(metode)):
     [svm, best_gamma[j], best_C[j]] =
trainData(dataTrain[j], y, metode[j])
```

Lampiran 3

Source code perbandingan metode optimasi fitur berdasar TF – NS.

```
import csv
import numpy as np
import scipy as sp
from pattern.db import Datasheet
from pattern.en import sentiment
from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer
from sklearn import preprocessing, cross_validation,
svm
from sklearn.cross_validation import StratifiedKFold
from sklearn.svm import SVC
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.decomposition import nmf
from sklearn.ensemble import ExtraTreesClassifier
import time

def getStopwords(namaTabelStopwords):
     with open(namaTabelStopwords, 'rb') as data:
          reader = csv.reader(data)
          stopword = []
          for row in reader:
               a        = row[0]
               stopword = stopword + [a]
          return stopword

def preprocessingTweet(tabelTweet, tabelStopword):
     metode = ['[TF-NS]', '[NMF]', '[ETC]']

     #CountVectorizer-NS
     CV  = CountVectorizer(stop_words =
tabelStopword).fit(tabelTweet)
     data = CV.transform(tabelTweet)
     X1  = data.toarray()

     #pakai dimension reduction NMF
     W, H = nmf._initialize_nmf(X1, 250)
     scaler    =
preprocessing.StandardScaler().fit(W)
     X2   = scaler.transform(W)

     scaler = preprocessing.StandardScaler().fit(X2)
     X3  = scaler.transform(X2)

     #pakai feature selection ExtraTrees
     clf = ExtraTreesClassifier(n_estimators=10,
max_features='auto', random_state=0,
compute_importances=True)
```

(Lanjutan)

```
     X4 = clf.fit(X1, y).transform(X1)

     print "Dimensi Data Train TF-NS: ",X1.shape
     print "Dimensi Data Train NMF:", X3.shape
     print "Dimensi Data Train ETC:", X4.shape
     dataTrain =    [X1, X3, X4]

     return [metode, dataTrain]

def trainData(arrayInput, arrayNilaiSentimen,
metode):

     tic = time.clock()
     ###Loose Grid-Search
     C_power   = np.arange(1, 9, 1)
     gamma_power = np.arange(-16, -4, 1)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a         = grid.fit(arrayInput,
arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_C_idx  = np.nonzero(C_power ==
best_C)[0][0]
     best_gamma  =
np.log2(grid.best_params_.values()[1])
     best_gamma_idx = np.nonzero(gamma_power ==
best_gamma)[0][0]

     print "best C : ",best_C, "best gamma : ",
best_gamma

     # Extracting Scores
     score_dict = grid.grid_scores_
     scores = [x[1] for x in score_dict]
     scores = np.array(scores).reshape(len(C_range),
len(gamma_range))

     ###Fine Grid-Search
     C_power   = np.arange(best_C-1.5, best_C+1,
0.25)
     gamma_power = np.arange(float(best_gamma)-1,
best_gamma+1, 0.25)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
```

(Lanjutan)

```
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a         = grid.fit(arrayInput,
arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_C_idx  = np.nonzero(C_power ==
float(str(best_C)))[0][0]
     best_gamma  =
np.log2(grid.best_params_.values()[1])
     best_gamma_idx = np.nonzero(gamma_power ==
float(str(best_gamma)))[0][0]

     print "best C : ",best_C, "best gamma : ",
best_gamma
     # Extracting Scores
     score_dict = grid.grid_scores_
     scores = [x[1] for x in score_dict]
     scores = np.array(scores).reshape(len(C_range),
len(gamma_range))

     toc = time.clock()
     print "runtime metode ",metode, ": ", toc-tic, "
detik."

     return [a, best_gamma, best_C]


#load tweet bahasa indonesia
with open('data_reinforce.csv', 'rb') as data:
     reader    = csv.reader(data)
     y    = []
     tweet     = []

     for row in reader:
          if row[7] == '-1':
               y    = np.append(y,[-1.0])
               tweet     = np.append(tweet,[row[2]])

          if row[7] == '1':
               y    = np.append(y, [1.0])
               tweet     = np.append(tweet, [row[2]])


stopword         = getStopwords('stopwords.csv')
[metode, dataTrain]   = preprocessingTweet(tweet,
stopword)
```

(Lanjutan)

```
best_gamma              = np.zeros(len(metode))
best_C                  = np.zeros(len(metode))


#compute Loose Grid and Fine Grid
for j in np.arange(len(metode)):
     [svm, best_gamma[j], best_C[j]] =
trainData(dataTrain[j], y, metode[j])
```

Lampiran 4

Source code perbandingan metode optimasi fitur berdasar TFIDF – S.

```
import csv
import numpy as np
import scipy as sp
from pattern.db import Datasheet
from pattern.en import sentiment
from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer
from sklearn import preprocessing, cross_validation,
svm
from sklearn.cross_validation import StratifiedKFold
from sklearn.svm import SVC
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.decomposition import nmf
from sklearn.ensemble import ExtraTreesClassifier
import time

def getStopwords(namaTabelStopwords):
     with open(namaTabelStopwords, 'rb') as data:
          reader     = csv.reader(data)
          stopword = []
          for row in reader:
               a = row[0]
               stopword = stopword + [a]
          return stopword

def preprocessingTweet(tabelTweet, tabelStopword):
     metode = ['[TFIDF-S]', '[NMF]', '[ETC]']

     #TFIDF-S
     CV = TfidfVectorizer(stop_words =
tabelStopword).fit(tabelTweet)
     data = CV.transform(tabelTweet)
     X1 = data.toarray()

     #pakai dimension reduction NMF
     W, H = nmf._initialize_nmf(X1, 250)
     scaler    =
preprocessing.StandardScaler().fit(W)
     X2   = scaler.transform(W)

     #pakai feature selection ExtraTrees
     clf = ExtraTreesClassifier(n_estimators=10,
max_features='auto', random_state = 0,
compute_importances=True)
     X3 = clf.fit(X2, y).transform(X2)
     print "Dimensi Data Train TFIDF-S:", X1.shape
     print "Dimensi Data Train NMF:", X2.shape
```

(Lanjutan)

```
     print "Dimensi Data Train ETC:", X3.shape

     dataTrain = [X1, X2, X3]

     return [metode, dataTrain]

def trainData(arrayInput, arrayNilaiSentimen,
metode):
     tic = time.clock()

     ###Loose Grid-Search
     C_power   = np.arange(-5, 17, 2)
     gamma_power = np.arange(-15, 5, 2)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a    = grid.fit(arrayInput, arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_gamma  =
np.log2(grid.best_params_.values()[1])

     ###Fine Grid-Search
     C_power   = np.arange(best_C-1.5, best_C+1,
0.25)
     gamma_power = np.arange(float(best_gamma)-1,
best_gamma+1, 0.25)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a = grid.fit(arrayInput, arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_gamma  =
np.log2(grid.best_params_.values()[1])

     toc = time.clock()
     print "runtime metode ",metode, ": ", toc-tic, "
detik."

     return [a, best_gamma, best_C]
```

(Lanjutan)

```
#load tweet bahasa indonesia
with open('data_reinforce.csv', 'rb') as data:
     reader    = csv.reader(data)
     y    = []
     tweet     = []

     for row in reader:
          if row[7] == '-1':
               y    = np.append(y,[-1.0])
               tweet     = np.append(tweet,[row[2]])

          if row[7] == '1':
               y    = np.append(y, [1.0])
               tweet     = np.append(tweet, [row[2]])


stopword         = getStopwords('stopwords.csv')
[metode, dataTrain]   = preprocessingTweet(tweet,
stopword)
best_gamma            = np.zeros(len(metode))
best_C                = np.zeros(len(metode))

for j in np.arange(len(metode)):
     [svm, best_gamma[j], best_C[j]] =
trainData(dataTrain[j], y, metode[j])
```

Lampiran 5

Source code perbandingan metode optimasi fitur berdasar *Binarization* – NS

```
import csv
import numpy as np
import scipy as sp
from pattern.db import Datasheet
from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer
from sklearn import preprocessing, cross_validation,
svm
from sklearn.cross_validation import StratifiedKFold
from sklearn.svm import SVC
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.decomposition import nmf
from sklearn.ensemble import ExtraTreesClassifier
import time

def getStopwords(namaTabelStopwords):
     with open(namaTabelStopwords, 'rb') as data:
          reader     = csv.reader(data)
          stopword = []
          for row in reader:
               a = row[0]
               stopword = stopword + [a]
          return stopword

def preprocessingTweet(tabelTweet, tabelStopword):
     metode = ['[Binarization-NS]', '[NMF]', '[ETC]']

     #TF-NS
     CV = CountVectorizer(stop_words =
tabelStopword).fit(tabelTweet)
     data = CV.transform(tabelTweet)
     X1 = data.toarray()

     #Binarization-NS (0 atau 1)
     biner     =
preprocessing.Binarizer(threshold=0.9).fit(X1) #data
> 0.9 menjadi bernilai 1, selain itu nilainya 0
     X2 = biner.transform(X1)

     #pakai dimension reduction NMF
     W, H      = nmf._initialize_nmf(X2, 250)
     scaler         =
preprocessing.StandardScaler().fit(W)
     X3        = scaler.transform(W)
     scaler = preprocessing.StandardScaler().fit(X3)
     X4 = scaler.transform(X3)
```

(Lanjutan)

```
     #pakai feature selection ExtraTrees
     clf = ExtraTreesClassifier(n_estimators=10,
max_features='auto', random_state=0,
compute_importances=True)
     X5 = clf.fit(X2, y).transform(X2)

     print "Dimensi Data Train Binarization-NS:",
X2.shape
     print "Dimensi Data Train NMF:", X4.shape
     print "Dimensi Data Train ETC:", X5.shape

     dataTrain = [X2, X4, X5]

     return [metode, dataTrain]

def trainData(arrayInput, arrayNilaiSentimen,
metode):
     tic = time.clock()
     ###Loose Grid-Search
     C_power   = np.arange(-5, 17, 2)
     gamma_power = np.arange(-15, 5, 2)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a    = grid.fit(arrayInput, arrayNilaiSentimen)
     best_C    =
np.log2(grid.best_params_.values()[0])
     best_gamma  =
np.log2(grid.best_params_.values()[1])

     ###Fine Grid-Search
     C_power   = np.arange(best_C-1.5, best_C+1,
0.25)
     gamma_power = np.arange(float(best_gamma)-1,
best_gamma+1, 0.25)
     C_range   = 2.0**C_power
     gamma_range = 2.0**gamma_power
     param_grid     = dict(gamma=gamma_range,
C=C_range)
     grid           = GridSearchCV(SVC(),
param_grid=param_grid, cv=StratifiedKFold(y =
arrayNilaiSentimen, n_folds = 5))
     a    = grid.fit(arrayInput, arrayNilaiSentimen)
     best_C = np.log2(grid.best_params_.values()[0])
     best_gamma  =
np.log2(grid.best_params_.values()[1])
```

(Lanjutan)

```
     toc = time.clock()
     print "runtime metode ",metode, ": ", toc-tic, "
detik."

     return [a, best_gamma, best_C]


#load tweet bahasa indonesia
with open('data_reinforce.csv', 'rb') as data:
     reader    = csv.reader(data)
     y    = []
     tweet     = []

     for row in reader:
          if row[7] == '-1':
               y    = np.append(y,[-1.0])
               tweet     = np.append(tweet,[row[2]])

          if row[7] == '1':
               y    = np.append(y, [1.0])
               tweet     = np.append(tweet, [row[2]])


stopword         = getStopwords('stopwords.csv')
[metode, dataTrain]   = preprocessingTweet(tweet,
stopword)
best_gamma            = np.zeros(len(metode))
best_C                = np.zeros(len(metode))


#compute Loose Grid and Fine Grid
for j in np.arange(len(metode)):
     [svm, best_gamma[j], best_C[j]] =
trainData(dataTrain[j], y, metode[j])
```