TUGAS AKHIR - KI141502

Aplikasi Deteksi Kejadian di Jalan Raya berdasarkan Data Twitter Menggunakan Metode Support Vector Machine

VESSA RIZKY OKTAVIA
NRP 5112100052

Dosen Pembimbing
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dini Adni Navastara, S.Kom, M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - KI141502

Aplikasi Deteksi Kejadian di Jalan Raya berdasarkan Data Twitter Menggunakan Metode Support Vector Machine

VESSA RIZKY OKTAVIA
NRP 5112100052

Dosen Pembimbing
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dini Adni Navastara, S.Kom, M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

FINAL PROJECT - KI141502

Event Detection Application on The Road based on Twitter Data using Support Vector Machine Method

VESSA RIZKY OKTAVIA
NRP 5112100052

Advisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dini Adni Navastara, S.Kom, M.Sc.

INFORMATICS DEPARTMENT
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

APLIKASI DETEKSI KEJADIAN DI JALAN RAYA BERDASARKAN DATA TWITTER MENGGUNAKAN METODE SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
VESSA RIZKY OKTAVIA
NRP 5112100052

Disetujui oleh Pembimbing Tugas Akhir:

1 Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.
NIP: 197512202001122002 (pembimbing 1)

2. Dini Adni Navastara, S.Kom, M.Sc.
NIP: 198510172015042001 (pembimbing 2)

SURABAYA
JANUARI, 2018

[Halaman ini sengaja dikosongkan]

APLIKASI DETEKSI KEJADIAN DI JALAN RAYA BERDASARKAN DATA TWITTER MENGGUNAKAN METODE SUPPORT VECTOR MACHINE

Nama Mahasiswa : Vessa Rizky Oktavia
NRP : 5112100052
Departemen : Informatika FTIK-ITS
Dosen Pembimbing I : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dosen Pembimbing II : Dini Adni Navastara, S.Kom, M.Sc.

ABSTRAK

Twitter adalah salah satu media sosial yang populer belakangan ini. Salah satu karakteristik penting dari Twitter adalah layanannya yang bersifat fleksibel yaitu dapat diakses di mana saja dan kapan saja. Sebagai contoh, saat terjadi suatu kecelakaan atau kemacetan, banyak pengguna Twitter yang mengirimkan informasi (tweets) tentang kejadian tersebut kepada Twitter. Hal ini memungkinkan dibuatnya sebuah sistem yang mendeteksi terjadinya kecelakaan atau kemacetan dengan melakukan observasi kepada tweet yang masuk.

Dalam tugas akhir ini, tweet akan diambil menggunakan Twitter API dan dimasukkan ke dalam sebuah database. Selanjutnya, akan dilakukan preproses yang meliputi stemming, penghapusan stopwords, dan tokenizing. Selain itu, dilakukan juga labeling untuk menentukan kelas dari tweet (kecelakaan, kemacetan, atau lain-lain). Selanjutnya akan dilakukan ekstraksi fitur agar fitur dari setiap tweet dapat menjadi input dalam proses klasifikasi. Untuk mengklasifikasikan tweet, diimplementasikan sebuah metode klasifikasi Support Vector Machine dan parameter regularisasi berupa variabel nu.

Model klasifikasi yang dibangun awalnya memberikan nilai akurasi 95,15%. Uji coba dilakukan dengan mengubah kernel dan

parameter nu untuk menghasilkan akurasi yang terbaik. Berdasarkan hasil uji coba yang telah dilakukan, didapatkan hasil terbaik dari sistem dengan akurasi 96,25% dengan klasifikasi menggunakan metode SVM dengan menggunakan Kernel Sigmoid dan parameter nu sebesar 0,2.

Kata kunci: Twitter, deteksi kejadian, Support Vector Machine, kecelakaan, kemacetan.

EVENT DETECTION APPLICATION ON THE ROAD BASED ON TWITTER DATA USING SUPPORT VECTOR MACHINE METHOD

Name	**: Vessa Rizky Oktavia**
NRP	**: 5112100052**
Major	**: Informatics Department, FTIK-ITS**
Advisor I	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Advisor II	**: Dini Adni Navastara, S.Kom, M.Sc.**

ABSTRACT

Twitter is one of the most popular social media lately. One of the important characteristics of Twitter is its flexible service that can be accessed anywhere and anytime. For example, when an accident or traffic jam occurs, many Twitter users are sending tweets about the event to the Twitter. This allows the creation of a system that detects accident or congestion by observing the incoming tweets.

In this thesis, tweet will be taken by using Twitter API and put into a database. Next, a preprocess will be done that includes stemming, stopwords removal, and tokenizing. In addition, there is also a labeling to determine the class of tweets (accidents, congestion, or others). Furthermore, feature extraction will be performed so that the features of each tweet can be the input to perform the classification process. To classify tweets, used a Support Vector Machine classification method andnu variable as a regularization parameters.

The classification model that was originally built gave an accuracy of 95.15%. The test is done by changing the kernel and the nu parameters to produce the best accuracy. Based on result of experiment which have done, the best result from system is claimed

with accuracy 96,25% by using classification using SVM method using Sigmoid Kernel and the number of parameter nu is 0,2.

Keywords: Twitter, event detection, Support Vector Machine, accidents, congestion.

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat-Nya kepada penulis, sehingga Tugas Akhir yang berjudul "***Aplikasi Deteksi Kejadian di Jalan Raya berdasarkan Data Twitter Menggunakan Metode Support Vector Machine***" ini dapat selesai walaupun memakan waktu yang sangat banyak.

Dalam pengerjaan Tugas Akhir ini penulis mendapatkan bantuan, dorongan dan juga motivasi dari berbagai pihak. Pada kesempatan kali ini, penulis ingin mencurahkan rasa terima kasih dan ungkapan syukur yang mendalam karena telah dipertemukan dengan :

1. Yuyun Wijayanti selaku ibu penulis yang selalu mengingatkan dan memberi motivasi dalam pengerjaan Tugas Akhir, Meidijas Yoedianto selaku ayah penulis yang senantiasa memberi masukan dan ilmu mengenai proses penyusunan Buku Tugas Akhir, dan keluarga penulis yang selalu memberikan doa dan dukungan kepada penulis dalam proses pengerjaan Tugas Akhir ini.
2. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. dan Ibu Dini Adni Navastara, S.Kom, M.Sc. sebagai dosen pembimbing yang telah memberikan ilmu dan kesabarannya serta senantiasa memantau kinerja penulis dalam mengerjakan Tugas Akhir ini.
3. Linggar, Miftah, Ilmi, Risyanggi, dan Hendro yang mau berbagi pengalaman, berdiskusi dengan penulis, memberikan bantuan dan motivasi dalam berbagai hal.
4. Eric, Reva, Maya, Rheza, dan Arief yang selalu mengingatkan penulis agar tidak patah semangat dan selalu berjuang untuk menyelesaikan Tugas Akhir ini.
5. Serta semua pihak yang tidak dapat disebutkan disini yang telah memberikan bantuan dalam pengerjaan Tugas Akhir ini.

Penulis menyadari bahwa dalam pengerjaan Tugas Akhir ini terdapat banyak kekurangan. Semua kekurangan itu berasal murni dari penulis. Oleh karena itu, kritik serta saran akan sangat bermanfaat untuk penyempurnaan Tugas Akhir secara lebih lanjut. Akhir kata, penulis meminta maaf bila penulisan laporan Tugas Akhir ini masih banyak kekurangan. Semoga hasil dari Tugas Akhir ini dapat memberikan manfaat kepada pembaca.

Surabaya, Januari 2018

Vessa Rizky Oktavia

DAFTAR ISI

LEMBAR PENGESAHAN vii
ABSTRAK ix
ABSTRACT xi
KATA PENGANTAR xiii
DAFTAR ISI xv
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Permasalahan 2
1.3. Batasan Permasalahan 2
1.4. Tujuan 3
1.5. Manfaat 3
1.6. Metodologi 3
1.7. Sistematika Penulisan 5
BAB II DASAR TEORI 7
2.1. Twitter 7
2.2. Twitter API 8
2.3. Kecelakaan 8
2.4. Kemacetan 9
2.5. Preproses, *Labeling* dan Ekstraksi Fitur 9
2.6. Support Vector Machine 11
2.7. *Kernel* 13
2.8. Parameter Regularisasi 15
BAB III ANALISIS DAN PERANCANGAN SISTEM 17
3.1. Perancangan Data 17
3.1.1. Perancangan Data Masukan 17
3.1.2. Perancangan Data Luaran 17
3.2. Perancangan Sistem 18
3.3. Perancangan Preproses Data, *Labeling* dan Ekstraksi Fitur Sebelum Melakukan Klasifikasi 19
3.4. Perancangan Metode Klasifikasi dengan Support Vector Machine 24

3.5. Perancangan Tampilan Statistik Hasil Klasifikasi pada Halaman *Web*.....25
BAB IV IMPLEMENTASI..... 27
4.1. Lingkungan Implementasi.....27
4.1.1. Perangkat Keras.....27
4.1.2. Perangkat Lunak.....27
4.2. Implementasi Modul Pengambilan Data 28
4.3. Implementasi Modul Preproses dan Pemberian Label 31
4.4. Implementasi Modul Penghitungan IDF 33
4.5. Implementasi Modul Penghitungan TF-IDF 35
4.6. Implementasi Modul Klasifikasi SVM.....36
4.7. Implementasi Modul Pembuatan Matriks.....41
4.8. Implementasi Pemuatan Hasil ke Dalam Halaman *Web* 41
BAB V PENGUJIAN DAN EVALUASI 45
5.1. Lingkungan Uji Coba 45
5.2. Data Uji Coba.....45
5.3. Skenario Uji Coba 47
5.3.1. Skenario Uji Coba Performa *Kernel* pada Metode SVM 47
5.3.2. Skenario Uji Coba Akurasi Klasifikasi SVM dengan Perubahan Nilai Parameter *Nu*.....48
5.4. Hasil Uji Coba 48
5.4.1. Hasil Uji Coba Performa *Kernel* pada Metode SVM 48
5.4.2. Hasil Uji Coba Akurasi Klasifikasi SVM dengan Perubahan Nilai Parameter *Nu* 49
5.5. Analisis Hasil Uji Coba 49
BAB VI KESIMPULAN DAN SARAN..... 53
6.1. Kesimpulan.....53
6.2. Saran.....53
DAFTAR PUSTAKA..... 55
LAMPIRAN 57
BIODATA PENULIS..... 61

DAFTAR GAMBAR

Gambar 2.1 Contoh Tweet 7
Gambar 2.2 Contoh Kecelakaan Mobil di Jalan Raya 9
Gambar 2.3 Contoh Kemacetan di Jalan Raya 10
Gambar 2.4 Ilustrasi SVM 12
Gambar 2.5 Ilustrasi *Multiclass* SVM 13
Gambar 2.6 Transformasi *Input Space* ke *Feature Space* 14
Gambar 3.1 Arsitektur Umum Sistem 18
Gambar 3.2 Contoh *Tweet* untuk Masing-Masing Kelas 20
Gambar 3.3 Tahapan Preproses, *Labeling*, dan Ekstraksi Fitur 21
Gambar 3.4 Proses Klasifikasi Sistem 22
Gambar 3.5 Macam-Macam *Kernel* dan Nilai Parameter *Nu* untuk Uji Coba 23
Gambar 3.6 Memuat Data ke Dalam Halaman *Web* 25
Gambar 4.1 Halaman *Web* yang Menampilkan Akurasi Awal Sistem 42
Gambar 4.2 Halaman *Web* yang Menampilkan Akurasi Sistem dan Hasil Cross Validation 42
Gambar 4.3 Hasil Statistik Kejadian 43
Gambar 5.1 Grafik Perubahan Akurasi SVM Terhadap Nilai Parameter *Nu* 51
Gambar 5.2 *Confusion Matrix* pada Skenario Terbaik *Nu* = 0,2 51
Gambar 5.3 *Confusion Matrix* pada Skenario Terburuk *Nu* = 0,9 51

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 4.1 Spesifikasi perangkat keras 27
Tabel 5.1 Contoh Tweet, Hasil Preproses, dan Labelnya..... 46
Tabel 5.2 Hasil Uji Coba Akurasi Metode Klasifikasi..... 48
Tabel 5.3 Daftar Akurasi dengan Mengubah *Nu*..... 49
Tabel 6.1 Hasil Uji Coba Akurasi Metode Klasifikasi SVM menggunakan Kernel Sigmoid dengan 10-Fold *Cross Validation* 57

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Twitter merupakan sebuah media sosial yang kini tren di masyarakat. Pada tahun 2017, jumlah pengguna aktif twitter mencapai 330 juta orang dan 80% pengguna aktif di perangkat mobile [1]. Fitur utama dari twitter yaitu *tweet* di mana pengguna dapat berkomunikasi dengan pengguna lainnya maupun menceritakan apapun yang pengguna inginkan. Di dalam Twitter, terdapat sebuah fitur yaitu *follow*, di mana user dapat mengikuti halaman akun user lain dan sebaliknya dapat diikuti oleh akun pengguna yang lain. Dengan menggunakan fitur *follow*, pengguna bebas memilih akun mana saja yang ingin mereka lihat tweetnya setiap hari. Sehingga, semakin banyak akun yang berusaha untuk membuat tweet yang menarik agar mendapatkan banyak akun yang mengikutinya atau disebut dengan *follower*.

Tingginya popularitas media sosial Twitter menyebabkan layanan ini telah dimanfaatkan untuk berbagai keperluan dalam berbagai aspek, misalnya sebagai sarana protes, kampanye politik, sarana pembelajaran, dan sebagai media komunikasi darurat. Twitter juga dihadapkan pada berbagai masalah dan kontroversi seperti masalah keamanan dan privasi pengguna, gugatan hukum, dan penyensoran. Hal ini menyebabkan banyak instansi maupun komunitas peduli lingkungan yang akhirnya membuat akun Twitter untuk menyampaikan berita tentang kejadian yang telah terjadi seperti kecelakaan maupun kemacetan. Twitter bersifat fleksibel sehingga membuat twitter menjadi media yang menarik digunakan untuk berbagai metode *event detection*. Data *tweet* yang didapatkan dari API twitter, bisa menjadi dasar untuk mendeteksi terjadinya sebuah event [2].

Oleh karena itu, pada Tugas Akhir ini dibuat sebuah aplikasi deteksi kejadian di jalan raya berdasarkan data twitter menggunakan metode Support Vector Machine. Sebuah event yang terdeteksi akan diklasifikasikan sebagai kecelakaan, kemacetan, atau lain-lain. Apabila terdeteksi suatu event, maka sistem akan mengklasifikasikan event tersebut dan mengambil statistik dari hasil klasifikasinya lalu ditampilkan ke dalam *web*.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan *preprocessing* terhadap data twitter yang didapat?
2. Bagaimana melakukan ekstraksi fitur pada *tweet* yang didapatkan?
3. Bagaimana mengimplementasikan metode klasifikasi yang mampu menentukan *tweet* yang memberikan informasi mengenai event yang sedang terjadi?
4. Bagaimana cara pengukuran untuk mengevaluasi kinerja sistem?
5. Bagaimana menampilkan hasil klasifikasi data ke dalam bentuk statistik kejadian?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Dataset *tweet* yang digunakan adalah *tweet* pada tahun 2017 yang dituliskan menggunakan Bahasa Indonesia.
2. Event yang terjadi berada di wilayah Indonesia, khususnya Surabaya.
3. Diasumsikan event yang dideteksi merupakan salah satu event kecelakaan atau kemacetan.
4. *Tweet* yang diambil berasal dari akun DISHUB SURABAYA (sits_dishubsby), infosurabaya

(infosurabaya), dan Radio Suara Surabaya (e100ss), serta beberapa akun pengguna twitter pada umumnya.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat sistem yang mampu mendeteksi terjadinya kejadian di jalan raya sebagai kecelakaan, kemacetan, dan lain-lain berdasarkan data twitter menggunakan metode *Support Vector Machine*.

1.5. Manfaat

Manfaat dari pembuatan tugas akhir ini adalah para peneliti dapat melihat perbedaan performa klasifikasi data teks dengan menggunakan metode *Support Vector Machine* dari sisi *kernel* dan parameter regularisasi yang menggunakan variabel *nu* agar dapat menjadi referensi dalam penelitian selanjutnya.

1.6. Metodologi

Tahap yang dilakukan untuk menyelesaikan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

Penyusunan proposal ini merupakan tahap awal dalam pengerjaan Tugas Akhir. Proposal tugas akhir ini berisikan penjelasan tentang apa yang akan dikerjakan dalam tugas akhir, menggunakan metode apa saja, serta apa hasil yang diharapkan. Proposal tugas akhir ini terdiri atas latar belakang, rumusan masalah, batasan masalah, serta tujuan dan manfaat dari tugas akhir. Selain itu dijelaskan juga tinjauan pustaka yang digunakan untuk referensi dalam pembuatan tugas akhir.

2. Studi Literatur

Pada tahap studi literatur ini, akan dipelajari berbagai referensi yang dibutuhkan dalam pembuatan tugas akhir, seperti Twitter

API, *library* apa saja yang digunakan dalam bahasa pemrograman python, algortima SVM, dan WIX *web designer*.

3. Analisis dan Desain Perangkat Lunak

Pada tahap ini akan dilakukan perancangan terhadap sistem yang akan dibuat. Bagaimana pembagian sistem menjadi komponen-komponen yang saling berinteraksi. Selain itu juga dilakukan analisis terhadap dataset yang didapatkan, apa saja tahap-tahap *preprocessing* yang perlu dilakukan.

4. Implementasi Perangkat Lunak

Aplikasi ini akan menggunakan bahasa pemrograman python. Untuk komunikasi dengan API Twitter akan digunakan melalui modul TwitterOAuth. Metode klasifikasi SVM akan menggunakan modul sklearn. Sementara untuk menyediakan hasil klasifikasi dengan web akan digunakan bahasa pemrograman HTML.

5. Pengujian dan Evaluasi

Ada 2 evaluasi yang akan diterapkan pada tugas akhir ini, yaitu:

a. Evaluasi *kernel* pada metode klasifikasi Support Vector Machine

Evaluasi ini akan dilakukan dengan melihat akurasi (%) dari hasil klasifikasi menggunakan beberapa jenis *kernel* pada metode SVM yang akan diuji coba. Berapa banyak *tweet* yang terklasifikasi sesuai dengan kelas seharusnya. Data pengujian ini diambil dari data historis *tweet* yang telah diambil.

b. Evaluasi nilai parameter *nu* yang optimal untuk mencapai akurasi yang maksimal

Evaluasi ini akan dilakukan dengan melihat akurasi (%) dari metode SVM yang menggunakan nilai parameter *nu* yang berbeda-beda. Kemudian, dilakukan analisis untuk melihat perbedaan antara nilai *nu* yang menghasilkan akurasi paling besar dan paling kecil.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika sebagai berikut:

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, batasan permasalahan, tujuan, metodologi, dan sistematika penulisan Buku Tugas Akhir.

2. Bab II Tinjauan Pustaka

Bab ini berisi tentang teori yang digunakan dan diimplementasikan pada Tugas Akhir. Teori-teori tersebut mencakup konsep dasar serta algoritma yang digunakan untuk menyelasikan permasalahan.

3. Bab III Analisis dan Perancangan Sistem

Bab ini berisi gambaran perangkat lunak secara umum. Pada bab ini dijelaskan tahapan-tahapan yang dijalankan oleh aplikasi.

4. Bab IV Implementasi

Bab ini berisi hasil implementasi rancangan aplikasi berupa kode program dalam bahasa pemrograman python.

5. Bab V Uji Coba dan Evaluasi

Bab ini berisi hasil evaluasi aplikasi dengan menggunakan aplikasi yang dibangun. Juga disertakan analisi dari hasil evaluasi perangkat lunak.

6. Bab VI Kesimpulan dan Saran

Bab ini merupakan penjelasan berupa hasil akhir yang dapat ditarik dari keselurahan proses dan percobaan Tugas Akhir. Selain itu di bab ini juga terdapat saran-saran yang berisi hal-hal yang dapat diperbaiki dan dikembangkan.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini.

2.1. Twitter

Twitter adalah media sosial yang menawarkan layanan pada penggunanya untuk dapat berbagi momen melalui teks, gambar, dan video. Twitter menyuguhkan sebuah pertanyaan kepada penggunanya, yaitu "*What's happening*?" (Apa yang sedang terjadi?) yang dapat dijawab dalam bentuk kalimat yang disebut *tweet* yang contohnya dapat dilihat pada Gambar 2.1. Pengguna dapat menjawab pertanyaan tersebut menggunakan tidak lebih dari 140 karakter sampai pada November 2017 akhirnya diupdate menjadi 280 karakter yang menggunakan bahasa selain Bahasa Korea, Jepang, dan Mandarin.

Gambar 2.1 Contoh Tweet

Salah satu aspek penting dari Twitter adalah karakteristiknya yang bersifat historis. Sebagai contoh, saat terjadi kemacetan atau kecelakaan, banyak pengguna mengirim jawaban ke twitter (*tweet*) yang berhubungan dengan dua kejadian tersebut. Hal ini memungkinkan dilakukan deteksi terhadap kejadian dengan menginspeksi berbagai *tweet* yang masuk ke Twitter.

2.2. Twitter API

Application Programming Interface (API) menspesifikasikan komponen perangkat lunak dalam bentuk fungsi-fungsi, input, output, dan juga tipe data dari keduanya [3]. Untuk mempermudah interaksi antara pengembang dan layanan Twitter, Twitter telah menyediakan API yang bisa diakses secara publik melalui internet.

Dalam tugas akhir ini, layanan Twitter API yang digunakan yaitu REST API. REST API adalah layanan Twitter API yang tidak memerlukan koneksi HTTP persisten. REST API digunakan untuk mengambil data historis dari twitter melalui REST API dan *Search* API.

2.3. Kecelakaan

Kecelakaan merupakan kejadian yang menyebabkan orang celaka, yang mana pada konteks tugas akhir ini terjadi di jalan raya seperti pada Gambar 2.2. Kecelakaan tidak hanya terjadi di jalan raya, bisa juga terjadi di jalan di perumahan, di gang-gang sempit, bahkan sampai ke halaman belakang rumah seseorang. Biasanya, kecelakaan diakibatkan kelalaian pengemudi, dalam kasus ini yang terjadi di jalan raya.

Seringkali kecelakaan disebabkan oleh keteledoran pengemudi yang seharusnya bisa dihindari seperti contohnya mengantuk ketika menyetir, mabuk, sakit mata, atau bahkan karena sedang menelepon saat mengemudi.

Gambar 2.2 Contoh Kecelakaan Mobil di Jalan Raya

2.4. Kemacetan

Kemacetan adalah situasi atau keadaan tersendatnya atau bahkan terhentinya lalu lintas yang disebabkan oleh banyaknya jumlah kendaraan melebihi kapasitas jalan. Kemacetan ini tidak hanya melibatkan pengguna jalan yang menggunakan kendaraan bermotor, bisa jadi kemacetan ini disangkut pautkan dengan adanya demonstrasi maupun kejadian yang menghambat lajunya lalu lintas seperti contohnya jalan berlubang, lampu lalu lintas, dan lain-lain.

Kemacetan sering terjadi di kota-kota besar seperti Jakarta, Bandung, dan Surabaya. Titik-titik di mana sering terjadinya kejadian macet ini juga tersebar di sejumlah jalan raya yang ada di Jawa Timur. Kemacetan tidak hanya dikarenakan kepadatan volume kendaraan yang ada di jalan, melainkan juga karena cuaca misalnya terjadi hujan yang mengakibatkan banjir.

2.5. Preproses, *Labeling* dan Ekstraksi Fitur

Preproses merupakan tahapan penting sebelum melakukan klasifikasi. Fitur di dalam sebuah *tweet* tidak dapat diekstrak dengan baik tanpa melalui preproses terlebih dahulu.

Gambar 2.3 Contoh Kemacetan di Jalan Raya

Pertama akan dilakukan proses *stemming* di mana setiap kata pada *tweet* akan dipecah menjadi kata dasar untuk memudahkan saat penghitungan nilai dari ekstraksi fitur. Sebagai contohnya kata ‘kemacetan’ akan menjadi kata ‘macet’ sehingga saat ada kata ‘macet’ di *tweet* yang lain, maka kedua kata ini akan dikelompokkan sebagai variabel yang sama.

Kemudian, menghilangkan kata-kata yang umum atau sering muncul di dalam sebuah kalimat (*stop words*) dari *tweet* seperti ‘dan’, ‘yang’, dan lain-lain. Daftar *stop words* dalam Bahasa Indonesia didapatkan dari *library* Sastrawi.

Langkah selanjutnya adalah melakukan normalisasi pada *tweet* dan menghilangkan beberapa karakter spesial ('[\'"+=!&?*^~#]'). Kemudian, dilakukan *labeling* untuk mengelompokkan *tweet* ke dalam 3 kelas yang terdiri dari kecelakaan, kemacetan, dan lain-lain.

$$W_{ij} = tf_{ij} \times \log(N / df_i) \quad \textbf{(2.1)}$$

Setelah *tweet* berhasil melalui *preprocessing*, maka selanjutnya akan dilakukan ekstraksi fitur yang merupakan kegiatan untuk mengambil sebuah fitur dari *tweet*.

Ekstraksi fitur ini dilakukan agar hasilnya dapat dianalisis untuk keperluan lebih lanjut dalam penelitian. Fitur yang akan diekstrak merupakan bobot TF-IDF yang mana *term frequency* menyatakan jumlah kata yang muncul dalam satu dokumen, dalam kasus ini satu *tweet*. Sedangkan *inverse document frequency* yang menyatakan seberapa pentingnya sebuah kata pada sekumpulan dokumen.

Untuk menghitung bobot (*W*) sebuah *tweet*, maka dapat digunakan rumus yang terdapat pada Formula 2.1. W_{ij} merupakan bobot kata t_j terhadap dokumen d_i yang mana dalam hal ini merupakan *tweet*. Lalu tf_{ij} melambangkan jumlah kemunculan kata t_j dalam dokumen d_i. *N* disini merupakan jumlah semua *tweet* yang ada dalam *database* dan df_i merupakan jumlah dokumen yang mengandung kata t_j.

2.6. Support Vector Machine

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada input space [4]. Pattern yang merupakan anggota dari dua buah kelas : +1 dan -1 dan berbagi alternatif garis pemisah *(discrimination boundaries)*. *Margin* adalah jarak antara *hyperplane* tersebut dengan pattern terdekat dari masing-masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM.

Data yang tersedia dinotasikan sebagai d x_i $\Re\in$ sedangkan label masing-masing dinotasikan $y_i \in\{-1,+1\}$ untuk i = 1,2,…,l , yang mana l adalah banyaknya data. Diasumsikan kedua kelas –1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi d, yang didefinisikan pada Formula 2.2.

Pattern $\vec{x}$ yang termasuk kelas -1 (sampel negatif) dapat dirumuskan sebagai *pattern* yang memenuhi pertidaksamaan yang

ditunjukkan pada Formula 2.3. Sedangkan *pattern* $\vec{x}$ yang *term*asuk kelas +1 (sampel positif) memenuhi pertidaksamaan yang ditunjukkan pada Formula 2.4.

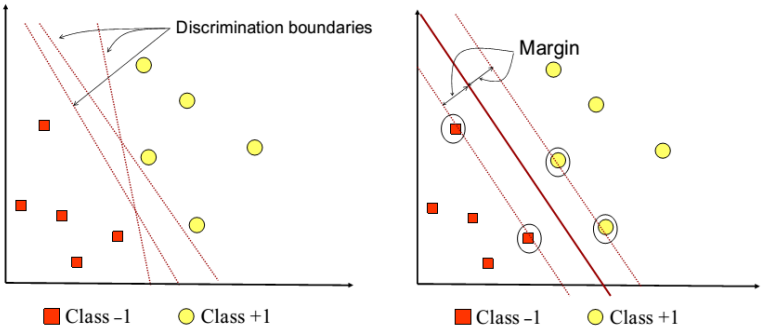
Pattern recognition dilakukan dengan mentransformasikan data pada *input space* ke ruang yang berdimensi lebih tinggi, dan optimisasi dilakukan pada ruang *vector* yang baru tersebut. Hal ini membedakan SVM dari solusi *pattern recognition* pada umumnya, yang melakukan optimasi parameter pada ruang hasil transformasi yang berdimensi lebih rendah daripada dimensi *input space*.

Model SVM seperti yang terlihat di Gambar 2.4 merepresentasikan data menjadi dua bagian dalam ruang, di mana titik terdekat dari dua bagian itu memiliki jarak sejauh-jauhnya. Data baru akan dipetakan ke ruang tersebut dan ditentukan kelasnya berdasarkan bagian mana dia berada. Dalam menggunakan metode SVM, bisa dilakukan klasifikasi non-linear dengan *kernel*, yaitu melakukan pemetaan input kepada dimensi yang lebih tinggi.

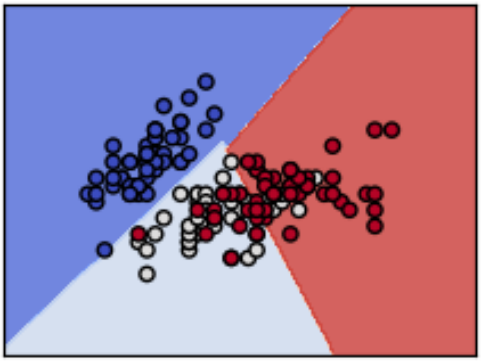
$$\vec{w} \cdot \vec{x} + b = 0 \quad \textbf{(2.2)}$$

$$\vec{w} \cdot \vec{x} + b \leq -1 \quad \textbf{(2.3)}$$

$$\vec{w} \cdot \vec{x} + b \geq +1 \quad \textbf{(2.4)}$$



Gambar 2.4 Ilustrasi SVM



Gambar 2.5 Ilustrasi *Multiclass* SVM

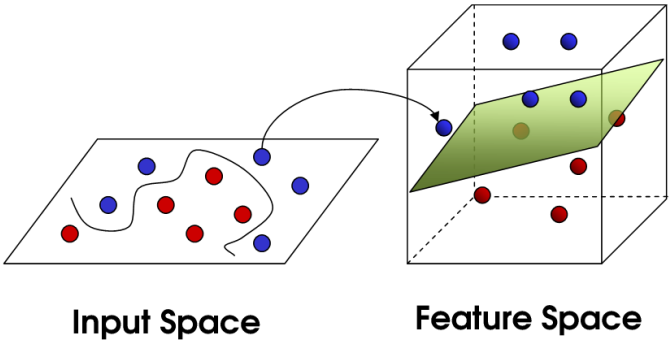
Karena pada dasarnya SVM adalah metode yang digunakan untuk mengklasifikasikan dua kelas [5], ada dua pilihan untuk mengimplementasikan *multiclass* SVM yang ilustrasinya dapat dilihat pada Gambar 2.5 yaitu dengan menggabungkan beberapa SVM biner atau menggabungkan semua data yang terdiri dari beberapa kelas ke dalam sebuah bentuk permasalahan optimal.

Namun pada pendekatan yang kedua permasalahan optimasi yang harus diselesaikan jauh lebih rumit. Metode yang umum digunakan untuk mengimplementasikan multiclass SVM dengan pendekatan yang pertama adalah Metode *One-Against-All* (satu lawan semua) yang mana dengan menggunakan metode ini, dibangun k buah model SVM biner (k adalah jumlah kelas) dan metode *One-Against-One* (satu lawan satu) yang dengan menggunakan metode ini, dibangun k(k-1)/2 buah model klasifikasi biner (k adalah jumlah kelas).

2.7. *Kernel*

Feature space dalam prosesnya biasanya memiliki dimensi yang lebih tinggi dari vektor input (*input space*) seperti yang terlihat pada Gambar 2.6. Hal ini akan mengakibatkan komputasi pada *feature space* akan menjadi sangat besar, karena ada

kemungkinan *feature space* akan memiliki jumlah feature yang tidak terhingga.



Gambar 2.6 Transformasi *Input Space* ke *Feature Space*

Untuk itu, pada SVM digunakan *kernel* sebagai cara untuk mengetahui fungsi transformasi yang tepat. Pemilihan fungsi *kernel* yang tepat adalah hal yang sangat penting. Karena fungsi *kernel* ini akan menentukan *feature space* di mana fungsi klasifier akan dicari [6]. Sepanjang fungsi *kernel*nya cocok, SVM akan beroperasi secara benar.

Kernel yang akan digunakan dalam klasifikasi SVM ini antara lain *kernel* linear, polynomial, RBF, dan sigmoid. Masing-masing *kernel* memiliki persamaan yang berbeda.

Kernel linear yang formulanya dapat dilihat pada Formula 2.5 adalah *kernel* yang paling sederhana dari semua fungsi *kernel*. *Kernel* ini biasa digunakan dalam kasus klasifikasi teks. *Kernel* Radial Basis Gaussian yang formulanya dapat dilihat pada Formula 2.6 adalah *kernel* yang umum digunakan untuk data yang sudah valid (available) dan merupakan default dalam tools SVM. *Kernel* Polynominal yang formulanya dapat dilihat pada Formula 2.7 adalah *kernel* yang sering digunakan untuk klasifikasi gambar sementara *Kernel* Sigmoid yang formulanya dapat dilihat pada Formula 2.8 adalah *kernel* yang sering digunakan untuk neural networks.

$$K\ (x_i, x) = \ x_i^T x \quad \textbf{(2.5)}$$

$$K\ (x_i, x) = \ (\gamma . x_i^T x + r)^p, \gamma > 0 \quad \textbf{(2.6)}$$

$$K\ (x_i, x) = \exp(\gamma |x_i - x|^2, \gamma > 0) \quad \textbf{(2.7)}$$

$$K\ (x_i, x) = \tanh(\gamma x_i^T + r) \quad \textbf{(2.8)}$$

2.8. Parameter Regularisasi

Parameter regularisasi adalah parameter yang menentukan besar penalti akibat kesalahan dalam klasifikasi data. Dalam metode SVM, parameter ini dikenal sebagai *C*. Parameter *C* bertugas mengkontrol *tradeoff* antara *margin* dan *classification error* yang dengan demikian membantu dalam meningkatkan akurasi output.

Semakin besar nilai *C*, semakin besar penalti yang dikenakan untuk tiap *classification error*. Nilai *C* berkisar dari mulai 0 sampai tak terhingga. Modifikasi untuk kasus ini adalah pengenalan parameter *nu* yang memiliki *range* antara 0 sampai 1 dan mewakili batas bawah dan atas pada jumlah contoh yang mendukung vektor dan berada pada sisi yang salah dari *hyperplane*.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas perancangan data, sistem, dan metode yang digunakan. Metode yang digunakan untuk klasifikasi *tweet* adalah metode support vector machine.

3.1. Perancangan Data

Pada bagian ini dijelaskan perancangan data yang akan digunakan dalam uji coba.

3.1.1. Perancangan Data Masukan

Data masukan dari sistem adalah data *tweet* yang didapatkan melalui API Twitter. Data *tweet* yang didapatkan memiliki atribut yaitu teks/isi dari *tweet* dan waktu kapan *tweet* dibuat.

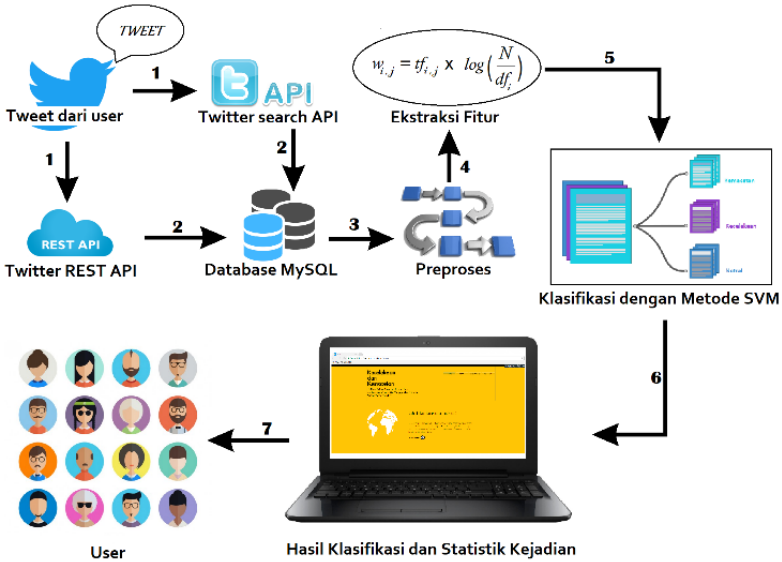
Data yang digunakan dalam tugas akhir ini adalah data *tweet* dari tanggal 1 Januari 2017 sampai dengan 1 November 2017. *Tweet* yang diambil dapat berasal dari akun-akun yang telah ditentukan pada batasan masalah dan akun dari pengguna Twitter yang memberikan *tweet* tentang kecelakaan atau kemacetan.

Jumlah *training* yang digunakan yaitu sebanyak 3649 tweet yang terdiri dari 1234 tweet dari Kelas Kecelakaan, 1190 tweet dari kelas kemacetan, dan 1225 tweet dari Kelas Lain-Lain. Kemudian diambil 30% dari data *training* untuk dijadikan data testing yaitu sejumlah 1095 tweet.

3.1.2. Perancangan Data Luaran

Data luaran yang didapatkan adalah hasil evaluasi akurasi model klasifikasi berdasarkan *cross-validation* dalam sistem mendeteksi kejadian di jalan raya yang merupakan kecelakaan dan kemacetan.

Selain itu, juga dihasilkan matriks yang merupakan hasil dari klasifikasi yang mengandung jumlah *error* dan jumlah kecocokan antara hasil klasifikasi yang salah dan benar dari tiap kelas sebagai perbandingan angka *nu* pada metode klasifikasi menggunakan SVM.



Gambar 3.1 Arsitektur Umum Sistem

3.2. Perancangan Sistem

Rancangan dari sistem pendeteksi kejadian kecelakaan dan kemacetan berdasarkan data twitter dapat dilihat pada Gambar 3.1.

Alur sistem dimulai saat modul crawl yang menggunakan *Search* API meminta data kepada Twitter dengan kriteria isi *tweet* yang mengandung kata ‘kecelakaan’ atau ‘kemacetan’ dan modul pengambilan data yang menggunakan REST API meminta data kepada Twitter dari halaman *timeline* milik beberapa akun yang khusus memberikan *tweet* mengenai kejadian-kejadian di jalan raya. Modul Crawl dibuat menggunakan bantuan *library* python *Tweet*. Semua data *tweet* yang telah dikumpulkan akan dimasukkan kedalam *database* MySQL untuk disimpan.

Selanjutnya, *tweet* yang telah masuk ke dalam *database* akan diproses kedalam modul preproses dan pemberian label, di mana modul ini akan melakukan preproses data sekaligus memberikan label pada setiap *tweet*. Preproses yang dilakukan meliputi

stemming dan penghapusan *stopwords* sekaligus normalisasi pada modul preproses dan *tokenizing* yang akan dilanjutkan di modul berikutnya. Modul preproses merupakan gabungan dari fungsi-fungsi yang diimplementasikan dari *library* Sastrawi yang merupakan *library* dengan bahasa pemrograman python khusus untuk preproses data text. Label pada setiap *tweet* dapat berbeda sesuai dengan kelasnya masing-masing yang terdiri dari 3 kelas, yaitu kemacetan, kecelakaan, dan lain-lain.

Selanjutnya data hasil preproses dan pemberian label akan masuk ke dalam modul untuk ekstraksi fitur yang terdiri dari TF (*term frequency*) dan IDF (*inverse document frequency*). Setelah didapatkan fiturnya, maka data akan diklasifikasi dengan model klasifikasi *Support Vector Machine* yang mana data akan dibagi menjadi dua bagian, *training* dan *testing*.

Model klasifikasi diimplementasikan dengan bantuan *library* scikit-learn. Pada saat melakukan klasifikasi, sekaligus akan dilakukan pengecekan terhadap akurasi sistem dengan menggunakan metode *cross-validation*.

Setelah selesai mengklasifikasikan data, hasil yang didapat dari setiap *tweet*nya akan masuk ke modul untuk pembuatan matriks untuk menghitung *tweet* mana saja yang mengalami penyimpangan label. Kemudian, hasil matriks yang berisi jumlah *tweet* ini bersama dengan semua *tweet*nya akan dimuat ke dalam halaman *web*.

3.3. Perancangan Preproses Data, *Labeling* dan Ekstraksi Fitur Sebelum Melakukan Klasifikasi

Rancangan dari tahapan preproses, *labeling*, dan ekstraksi fitur pada sistem dapat dilihat pada Gambar 3.3 sedangkan contoh *tweet* yang tergolong kemacetan, kecelakaan, dan lain-lain dapat dilihat pada Gambar 3.2.

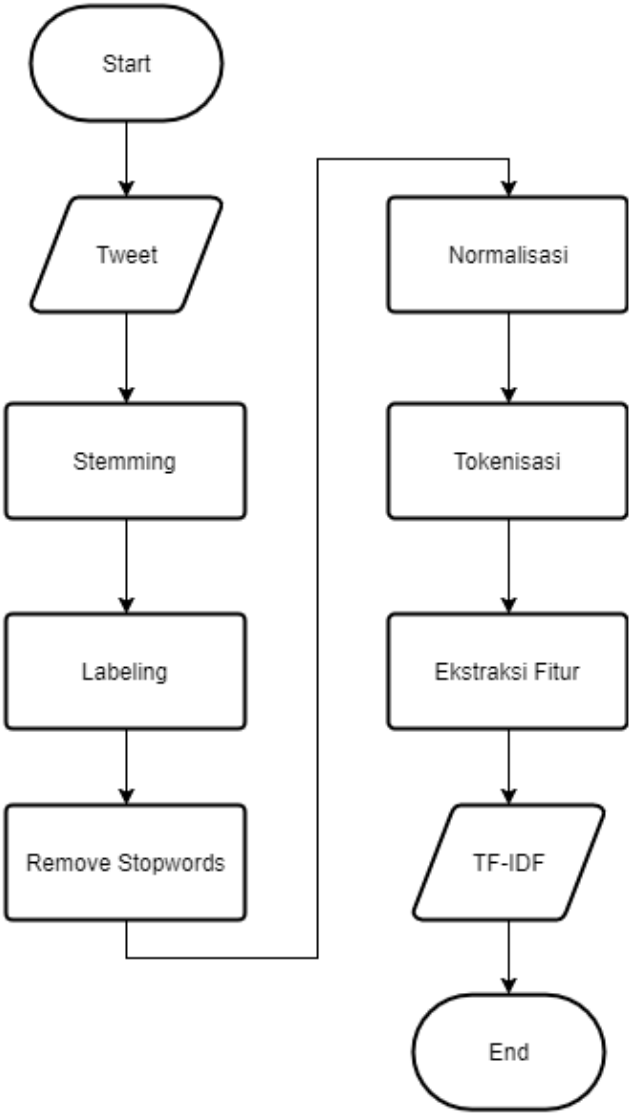
Tweet akan diproses sebagai text dan akan menjalani proses *stemming* di mana setiap kata pada *tweet* akan dipecah menjadi kata dasar untuk memudahkan saat penghitungan nilai dari ekstraksi

fitur. Pembentukan kata dasar ini menggunakan fungsi *stemming* dari *library* Sastrawi.

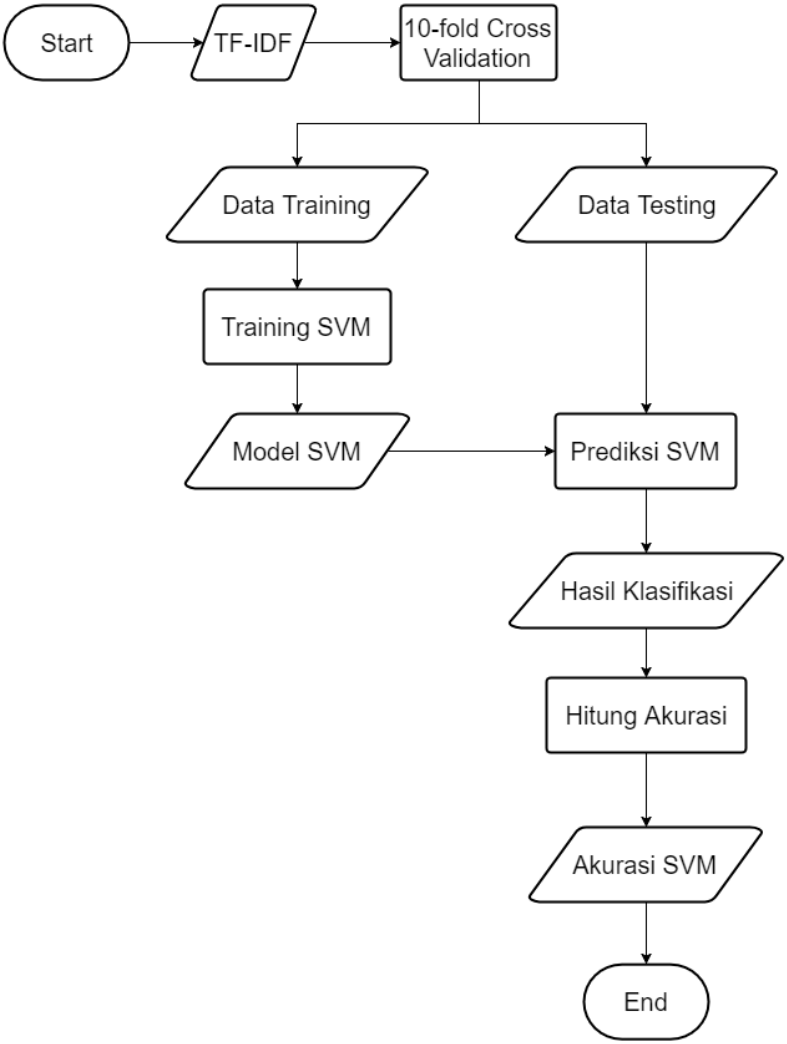
Lalu, dilakukan penghapusan *stopwords* dan normalisasi atau penghapusan tanda baca yang sering digunakan dalam text. Kemudian, dilakukan *labeling* untuk mengelompokkan *tweet* ke dalam 3 kelas yang terdiri dari kecelakaan, kemacetan, dan lain-lain.

Label didapatkan dengan menyeleksi kata-kata yang dapat mengategorikan *tweet* sebagai ‘kecelakaan’ seperti ‘celaka’, ‘tabrak’, ‘jatuh’ atau ‘kemacetan’ seperti ‘mogok’, ‘padat’, dan ‘macet’. Kata-kata yang diambil sebagai perwakilan setiap label adalah diambil berdasarkan pengamatan penulis terhadap pola dataset yang telah diambil.

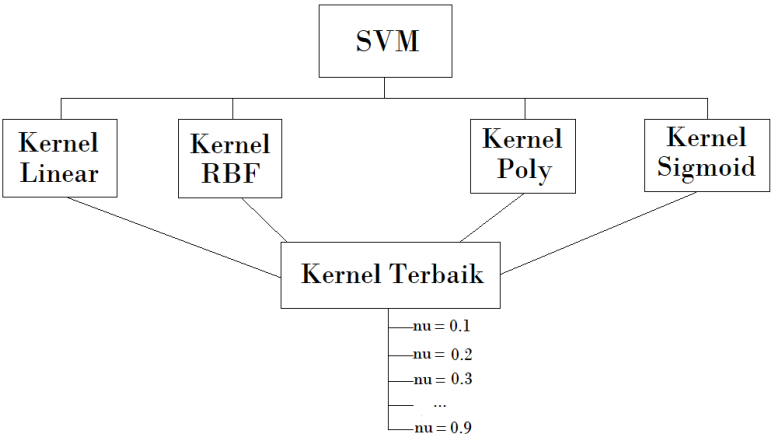
Gambar 3.2 Contoh *Tweet* untuk Masing-Masing Kelas



Gambar 3.3 Tahapan Preproses, *Labeling*, dan Ekstraksi Fitur



Gambar 3.4 Proses Klasifikasi Sistem



Gambar 3.5 Macam-Macam *Kernel* dan Nilai Parameter *Nu* untuk Uji Coba

Berdasarkan hasil pengamatan, didapatkan kata-kata yang sering digunakan di dalam sebuah *tweet* yang memiliki kecenderungan ke dalam Kelas Kecelakaan dan Kelas Kemacetan.

Kebanyakan *tweet* yang berhubungan dengan kecelakaan mengandung kata ‘celaka’, ‘tabrak’, dan ‘jatuh’ sedangkan untuk *tweet* yang berhubungan dengan kemacetan mengandung kata ‘mogok’, ‘padat’, dan ‘macet’. Kata-kata ini nantinya akan menjadi penentu kelas atau label dari *tweet* dan label ini nantinya akan menjadi *y* atau target pada saat klasifikasi.

Setelah *tweet* berhasil melalui *preprocessing*, maka selanjutnya akan dilakukan tokenisasi dan dilanjutkan dengan ekstraksi fitur yang merupakan kegiatan untuk mengambil sebuah fitur dari *tweet*.

Ekstraksi fitur ini dilakukan agar hasilnya dapat dianalisis untuk keperluan lebih lanjut dalam penelitian. Fitur yang dimaksud disini yaitu nilai TF-IDF dari semua *tweet*. TF-IDF inilah yang nantinya akan menjadi input dalam proses klasifikasi sebagai *X* dalam dataset.

3.4. Perancangan Metode Klasifikasi dengan Support Vector Machine

Langkah-langkah yang harus dilakukan dalam klasifikasi dengan menggunakan SVM ini yang pertama adalah pelatihan *(training)* yaitu mempelajari pola-pola teks pada *tweet* menjadi model, sesuai acuan yang diberikan. Acuan berupa teks panduan yang berlabel kelas kemacetan, kecelakaan, atau tidak sama sekali.

Kemudian langkah kedua yaitu prediksi *(testing)* yaitu menggunakan model untuk mengklasifikasikan teks dan hasilnya merupakan prediksi dari klasifikasi. Hasil prediksi dapat mengandung bias maupun kesalahan.

Pada rancangannya, *tweet* dan label akan dibagi menjadi data training dan testing seperti pada Gambar 3.4. Kemudian setelah dilakukan pelatihan, dilanjutkan dengan proses prediksi yang mana akan menghasilkan akurasi sistem sesuai dengan metode yang digunakan yaitu SVM.

Selain itu, terdapat banyak *kernel* yang dapat digunakan saat melakukan klasifikasi menggunakan SVM, antara lain *kernel* linear, rbf, poly, dan sigmoid. Kemungkinan akurasi yang dihasilkan oleh masing-masing *kernel* dapat berbeda seperti yang dapat dilihat pada Gambar 3.5 tergantung dari seberapa efektif penggunaan *kernel* untuk dataset yang diinputkan.

Pada klasifikasi *tweet* ini, digunakan metode SVM yang ada di dalam *library* scikit-learn. Terdapat banyak jenis *library* SVM dengan berbagai macam *kernel* yang berbeda, di antaranya *library* SVC, SVR, dan NuSVC. SVC merupakan *library* untuk melakukan klasifikasi menggunakan metode SVM menggunakan parameter *C* sebagai parameter regularisasi, sedangkan untuk SVR adalah *library* untuk melakukan klasifikasi SVM secara regresi.

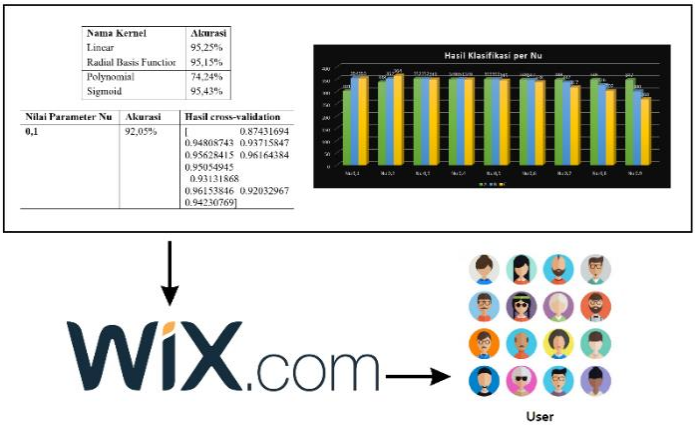
Sementara itu, *library* NuSVC adalah library untuk melakukan klasifikasi menggunakan metode SVM, namun dengan menggunakan parameter regularisasinya yaitu *nu*. Sehingga, untuk tugas akhir ini digunakan NuSVC karena untuk uji coba, parameter regularisasi yang akan digunakan berupa *nu* bukan parameter yang berupa *C*.

Jadi, selain mempertimbangkan pemilihan *kernel*, perlu diperhitungkan juga nilai parameter *nu* dalam penggunaan metode klasifikasi menggunakan *library* NuSVC ini sehingga dilakukan dua jenis uji coba.

3.5. Perancangan Tampilan Statistik Hasil Klasifikasi pada Halaman *Web*

Rancangan tampilan statistik hasil klasifikasi pada halaman *web* dapat dilihat pada Gambar 3.6. Di dalam *website* akan ditampilkan hasil akurasi klasifikasi SVM dari uji coba yang telah dilakukan. Pertama akan ditampilkan akurasi awal dari sistem yang menggunakan *kernel default* yaitu RBF dan nilai *nu default* yaitu 0,5 menggunakan *library* NuSVC.

Kemudian, akan ditampilkan hasil akurasi klasifikasi SVM dengan menggunakan *kernel* yang lainnya. Selanjutnya akan ditampilkan hasil akurasi dari masing-masing perubahan nilai parameter *nu*. Selain itu, hasil dari jumlah *tweet* yang diklasifikasikan secara tepat per kelasnya akan dibuat sebagai suatu statistik dan digolongkan per parameter *nu*.



Gambar 3.6 Memuat Data ke Dalam Halaman *Web*

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Dalam Bab IV ini akan dijelaskan algoritma untuk pemrograman dan masing-masing fungsi dari penggunaan *library* yang dibutuhkan dan implementasinya dalam program. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Python dan mengandung sedikit bahasa pemrograman lainnya seperti SQL Query.

4.1. Lingkungan Implementasi

Dalam implementasi algoritma digunakan perangkat-perangkat sebagai berikut:

4.1.1. Perangkat Keras

Spesifikasi perangkat keras yang digunakan saat implementasi ditunjukan pada Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat keras

Perangkat	Processor	Memori
MSI GL62 7RD	Intel Core i7-7700HQ	8 GB

4.1.2. Perangkat Lunak

Berikut perangkat lunak yang digunakan saat implementasi:

1. Windows 10
2. Thonny IDE
3. Sublime 3
4. Python 3.6 64-bit
5. XAMPP
6. *Library* Python numpy
7. *Library* Python scikit-learn
8. *Library* Python Tweepy
9. *Library* Python Sastrawi
10. *Library* Python xlsxwriter

4.2. Implementasi Modul Pengambilan Data

Hal pertama yang dilakukan adalah pengambilan data melalui 2 API, yaitu REST API dan *Search* API. REST API akan digunakan untuk mengambil semua *tweet* yang muncul di timeline lalu memberi label pada tiap *tweet*, sedangkan untuk *Search* API akan melakukan pencarian *tweet* yang mengandung kata-kata tertentu yang telah dikelompokkan menjadi kategori kemacetan atau kecelakaan. Program ini adalah bagian sistem yang langsung berhubungan dengan API Twitter.

Pertama kode program akan membuka sebuah koneksi persisten HTTP dengan REST API Twitter. Setiap *tweet* yang muncul pada home timeline akun Analisis Trafik, akan disimpan ke dalam *database* MySQL yang nantinya akan menuju ke implementasi selanjutnya untuk preproses *tweet*.

Tahapan yang harus dilakukan, pertama masukkan *library* yang diinginkan dengan menjalankan fungsi import yang diikuti dengan nama *library* seperti pada Kode Sumber 4.1 baris 1 sampai 4. Kemudian, secara khusus mengimport fungsi OauthHandler yang ada di *library* tweepy bagian authentication.

Masukkan *consumer key, consumer secret, access token* dan *access secret* untuk membuka koneksi HTTP dengan aplikasi yang telah didaftarkan di twitter ke dalam variabel yang diinginkan seperti pada Kode Sumber 4.2 baris 1 sampai 4. Kemudian, jalankan fungsi OauthHandler untuk mengotentikasi aplikasi yang telah didaftarkan di twitter dengan program python dan membuka koneksi dengan API twitter menggunakan tweepy.API seperti pada Kode Sumber 4.2 baris 5 sampai 7.

Langkah selanjutnya adalah membuka koneksi dengan *database* MySQL menggunakan pymysql.connect.

```
1. import tweepy
2. import time
3. import pymysql.cursors
4. from tweepy.auth import OauthHandler
```

Kode Sumber 4.1 Daftar *Library* untuk Pengambilan Data

```
1. consumer_key = 'iUPkpD33oAys46ZHkmPrOory4'
2. consumer_secret =
   'VocXqZFEoGIuWaU9S3wtWjbrlNFyiTOYLeEhP5muU
   GLTy1UVRE'
3. access_token = '864119721699889155-
   gSrUQviQ0ozQIn34sWS3e0wPBUB4c1h'
4. access_secret =
   'vDQHDZaHIZRnVUVw6K7qbHb6JYjonRqkxrwStHEIN
   TrFo'
5. auth = OAuthHandler(consumer_key, consumer_secret)
6. auth.set_access_token(access_token, access_secret)
7. api = tweepy.API(auth)
```

Kode Sumber 4.2 Membuka Koneksi dengan MySQL

Parameter penting yang diperlukan yaitu nama host, nama user, password (jika tidak ada maka bisa dikosongi), nama *database*, jenis charset, dan cursor class yang ingin digunakan seperti pada Kode Sumber 4.3 baris 1. Lalu, deklarasikan variabel untuk menggunakan cursor class yang diinginkan yaitu c sebagaimana pada Kode Sumber 4.3 baris 2 sampai 4. Disini digunakan class DictCursor karena yang dibutuhkan adalah cursor untuk mengambil data dalam bentuk dictionary yang nantinya akan memiliki beberapa parameter untuk kebutuhan preproses.

Untuk menggunakan REST API, fungsi yang digunakan adalah home_timeline, sedangkan untuk *Search* API, fungsi yang digunakan adalah *Search*. Untuk *Search* API yang membutuhkan parameter query kata atau frase yang akan dicari, maka dibutuhkan kata yaitu kemacetan dan kecelakaan. Dalam menjalankan ketiga kode ini, harus dilakukan satu persatu dalam setiap kali running dengan menggunakan REST API sekali, dilanjutkan dengan mengganti api.home_timeline menjadi api.*Search* untuk menggunakan *Search* API dengan query kemacetan, lalu dilanjutkan dengan pencarian query kecelakaan yang juga menggunakan *Search* API.

```
1. connection = pymysql.connect(host='localhost', user='root',
   password='', db='test', charset='utf8mb4',
   cursorclass=pymysql.cursors.DictCursor)
2. c = tweepy.Cursor(api.home_timeline,
   include_entities=True).items()
3. c = tweepy.Cursor(api.Search, q="kemacetan",
   include_entities=True).items()
4. c = tweepy.Cursor(api.Search, q="kecelakaan",
   include_entities=True).items()
```

Kode Sumber 4.3 Mengambil Data dari Twitter

Terakhir, membuat sebuah *infinite loop* seperti pada Kode Sumber 4.4 baris 1 sampai 28 untuk mengambil data dari twitter yang berupa *tweet* dengan atributnya yaitu *tweet*.id, *tweet*.text, dan *tweet*.created_at. Data yang telah didapatkan ini akan dimasukkan ke dalam *database* menggunakan bahasa pemrograman SQL. Untuk menjalankan query ke dalam *database*, harus digunakan connection.commit() agar data dapat disimpan ke dalam *database*.

```
1.  while True:
2.      try:
3.          tweet = c.next()
4.          tweet_text = tweet.text
5.          with connection.cursor() as cursor:
6.              sql = "INSERT INTO `tweet` (`Tweet_ID`,
                `TEXT`, `DATE`) VALUES (%s, %s, %s)"
7.              cursor.execute(sql, (tweet.id, tweet.text,
                tweet.created_at))
8.              connection.commit()
9.      except tweepy.TweepError:
10.         time.sleep(60 * 15)
11.         continue
12.     except StopIteration:
13.         break
```

Kode Sumber 4.4 Memasukkan *Tweet* ke *Database*

```
1. from Sastrawi.Stemmer.StemmerFactory import
   StemmerFactory
2. from
   Sastrawi.StopWordRemover.StopWordRemoverFactory
   import StopWordRemoverFactory
```

Kode Sumber 4.5 Memanggil *Library* Sastrawi

Dikarenakan twitter memiliki limit pada jumlah data yang diambil setiap harinya dan waktu untuk pengambilan data, maka diberikan exception untuk mengeksekusi program hanya setiap 15 menit sekali dan akan berhenti jika jumlah *tweet* yang diambil sudah mencapai limit.

4.3. Implementasi Modul Preproses dan Pemberian Label

Dalam implementasi ini akan dilakukan preproses terhadap *tweet* yang telah masuk ke dalam *database* sekaligus untuk memberikan label awal *tweet* sesuai dengan kelasnya masing-masing. Ada 2 langkah dalam preproses, stemming, yaitu menjadikan setiap kata pada *tweet* menjadi kata dasar, dilanjutkan dengan penghilangan stopwords, yaitu kata yang sering digunakan dalam sebuah kalimat seperti “dan”, “yang”, dan lain-lain.

Dibutuhkan 2 *library* yaitu pymysql.cursor untuk membuka koneksi dengan *database* MySQL sama seperti pada implementasi yang sebelumnya untuk pengambilan data, lalu *library* Sastrawi yang merupakan *library* python khusus untuk preproses dengan menggunakan Bahasa Indonesia dan pemanggilan *library* dilakukan sesuai dengan Kode Sumber 4.5 baris 1 dan 2.

```
1. factory = StemmerFactory()
2. stemmer = factory.create_stemmer()
3. stopfactory = StopWordRemoverFactory()
4. stopword = stopfactory.create_stop_word_remover()
```

Kode Sumber 4.6 Membuat Objek Stemmer dan Stopword Remover

```
1. word_1 = "celaka"
2. word_2 = "jatuh"
3. word_3 = "tabrak"
4. word_4 = "macet"
5. word_5 = "padat"
6. word_6 = "mogok"
```

Kode Sumber 4.7 Daftar *Term* Kelas Kecelakaan dan Kemacetan

```
 7. label = 0
 8. for row in result:
 9.     stemmed  = stemmer.stem(row['TEXT'])
10.     output = stopword.remove(stemmed)
11.     id = row['ID']
12.     if any(x in output for x in (word_1,word_2,word_3)):
13.         label = 1
14.         sql = "UPDATE `tweet` SET `LABEL` = %s,
            `PREPROCESSED` = %s WHERE `ID` = %s"
15.         cursor.execute(sql, (int(label),output,id))
16.     elif any(x in output for x in (word_4,word_5,word_6)):
17.         label = 2
18.         sql = "UPDATE `tweet` SET `LABEL` = %s,
            `PREPROCESSED` = %s WHERE `ID` = %s"
19.         cursor.execute(sql, (int(label),output,id))
20.     else:
21.         label = 0
22.         sql = "UPDATE `tweet` SET `LABEL` = %s,
            `PREPROCESSED` = %s WHERE `ID` = %s"
23.         cursor.execute(sql, (int(label),output,id))
```

Kode Sumber 4.8 Memberi Label untuk Setiap *Tweet*

Berikutnya adalah membuat objek untuk menjalankan fungsi yang ada pada *library* Sastrawi, dalam ini yang digunakan adalah stemmer dan stop_word_remover sesuai dengan Kode Sumber 4.6 baris 1 sampai 4.

```
24.  connection.commit()
25.  finally:
26.      connection.close()
```

Kode Sumber 4.9 Menyimpan Perubahan ke *Database* dan Menutup Koneksi ke *Database*

Selanjutnya adalah memberikan batasan terhadap *labeling*, dalam kasus ini digunakan 3 kata untuk kemacetan dan 3 kata untuk kecelakaan. Daftar kata dapat dilihat pada Kode Sumber 4.7 baris 1 sampai 6.

Kemudian, ambil semua data dari *database*, lalu gunakan loop untuk menjalankan stemming, menghapus stopwords, dan memberikan label pada setiap *tweet*nya sesuai dengan rule yang telah dibuat. Label 1 untuk kecelakaan, label 2 untuk kemacetan, dan sisanya label 0 untuk *tweet* yang tidak masuk ke dalam 2 kelas sebelumnya sesuai dengan Kode Sumber 4.8 baris 7 sampai 23.

Setelah semua query telah dijalankan, sertakan connection.commit() seperti pada Kode Sumber 4.9 baris 24 untuk menyimpan perubahan ke dalam *database* dan diakhiri dengan menutup koneksi MySQL yang dapat dilihat pada Kode Sumber 4.9 baris 25 dan 26.

4.4. Implementasi Modul Penghitungan IDF

Inverse Document Frequency adalah nilai yang didapatkan dari operasi logaritma berbasis 10 dari hasil pembagian antara jumlah dokumen yang ada dengan jumlah dokumen yang memuat tersebut. Jumlah dokumen atau *tweet* yang ada yaitu 3649 *tweet*. Untuk itu, dibuat sebuah fungsi yaitu contain_words pada Kode Sumber 4.10 baris 2 dan 3 untuk mengecek apakah dalam sebuah *tweet* mengandung *term* tertentu.

Selain itu, dibutuhkan *library math* yaitu pada Kode Sumber 4.10 baris 1 yang merupakan *library* bawaan dari python untuk melakukan berbagai operasi matematis, yang mana dalam kasus ini adalah logaritma.

```
1. Import math
2. def contains_word(s, w):
3.     return (' ' + w + ' ') in (' ' + s + ' ')
```

Kode Sumber 4.10 Membuat Fungsi untuk Menghitung Jumlah *Term* pada *Tweet* Menggunakan *Library Math*

```
1. sql = "SELECT `ID`,`term` FROM `words`"
2.     cursor.execute(sql)
3.     result = cursor.fetchall()
4.     for row in result:
5.         id = row['ID']
6.         term = row['term']
7.         sql = "SELECT `PREPROCESSED` FROM `tweet`"
8.         cursor.execute(sql)
9.         hasil = cursor.fetchall()
10.        idf=0
11.        for row in hasil:
12.            text = row['PREPROCESSED']
13.            return_value = contains_word(text,term)
14.            if return_value == 1:
15.                idf+=1
16.        if idf==0:
17.            idf = 1
```

Kode Sumber 4.11 Menghitung Frekuensi Dokumen

Selanjutnya akan dibuat loop untuk mengambil semua *term* yang ada pada semua *tweet* yang telah dipreproses seperti pada Kode Sumber 4.11 baris 4 sampai 9, kemudian pada setiap *term* akan dilakukan perulangan untuk mengecek jumlah dokumen/*tweet* yang memuat *term* tersebut seperti pada Kode Sumber 4.11 baris 10 sampai 15. Langkah berikutnya adalah menyiapkan variabel untuk menampung jumlah dokumen yang mengandung *term* tertentu yaitu variabel idf pada kasus ini. Variabel idf pada awalnya dianggap 0, kemudian seiring dengan berjalannya pengecekan, apabila sebuah *tweet* mengandung *term*

yang sedang dicek, maka variabel idf akan bertambah sesuai dengan jumlah dokumen yang memuat *term* tersebut seperti pada Kode Sumber 4.11 baris 16 sampai 18.

Namun, dikarenakan dapat terjadi eror yang dapat menyebabkan *term* tidak terdeteksi di dokumen manapun, maka diberikan kondisi lanjutan yang menganggap variabel idfnya sebagai 1 seperti seperti pada Kode Sumber 4.11 baris 19 dan 20.

Jumlah *tweet* yang merupakan angka tetap 3649 nantinya akan dibagi dengan jumlah dokumen yang memuat *term* (idf) lalu diambil hasil logaritma berbasis 10 menggunakan *library* math yang akan menghasilkan frekuensi dokumen invers per *term*nya seperti pada Kode Sumber 4.12 baris 1 dan 2.

Hasil dari implementasi idf ini dapat ditampung ke dalam file .txt sehingga memudahkan dalam pendeteksian eror maupun untuk penggunaan kembali jika ingin menggunakannya untuk operasi yang lain.

Setelah mendapatkan hasil idf setiap *term* pada dokumen yang telah dimasukkan ke dalam file idf.txt, selanjutnya akan dilakukan penghitungan frekuensi *term* (tf).

4.5. Implementasi Modul Penghitungan TF-IDF

Langkah pertama yang dilakukan pada penghitungan tf ini adalah sama seperti sebelumnya, yaitu membuat dua loop untuk melakukan iterasi pengambilan semua *term* yang ada pada semua *tweet*. Setelah itu, sediakan sebuah variabel untuk menampung banyaknya jumlah kata yang ada dalam sebuah *tweet* seperti pada Kode Sumber 4.13 baris 3.

```
1. log_idf = math.log10(3649 / idf)
2.     print(log_idf)
```

Kode Sumber 4.12 Menghitung Nilai IDF

```
3. number_of_words = len(text.split())
4. freq = text.count(term)
5. tf = float(freq) / number_of_words
```

Kode Sumber 4.13 Menghitung Nilai TF

```
1. import numpy as np
2. data = np.loadtxt("idf.txt", delimiter="\n")
3. tfidf = tf * data[kolom]
```

Kode Sumber 4.14 Menghitung TF-IDF

```
4. import xlsxwriter
5. workbook = xlsxwriter.Workbook('tfidf.xlsx')
6. worksheet = workbook.add_worksheet()
7. worksheet.write_number(baris, kolom, tfidf)
```

Kode Sumber 4.15 Memasukkan TF-IDF ke Dalam File Excel

Lalu, gunakan fungsi count untuk menghitung jumlah *term* yang ingin dihitung dari sebuah *tweet* seperti pada Kode Sumber 4.13 baris 4.

Setelah itu, hitung frekuensi *term* tiap kata dengan cara membagi jumlah *term* tertentu pada *tweet* dengan jumlah *term* pada *tweet* tersebut seperti pada Kode Sumber 4.13 baris 5.

Kemudian, untuk mendapatkan idf pada *file* idf.txt, dibutuhkan *library* numpy agar data di dalam *file* dapat ditampung menggunakan sebuah *array* seperti pada Kode Sumber 4.14 baris 1 sampai 3.

Selanjutnya, seiring dengan iterasi yang dilakukan per *term*nya, akan dihasilkan tf untuk setiap *term*, yang mana hasilnya jika dikalikan dengan tiap elemen pada *file* idf.txt akan menjadi TF-IDF seperti pada Kode Sumber 4.15 baris 4 sampai 7. Hasil TF-IDF ini akan dituliskan sebagai file excel agar mudah untuk dikonversikan menjadi file berformat .csv.

4.6. Implementasi Modul Klasifikasi SVM

Hasil dari modul sebelumnya adalah berupa matriks yang berisikan nilai TF-IDF dari semua *tweet* yang masing-masingnya diwakilkan oleh *term-term* yang terkandung di dalam *tweet*. File yang didapatkan berupa file dengan format .xlsx yang dapat dikonversikan menjadi berformat .csv (Comma-separated Values).

```
1. import numpy as np
2. data_train = np.loadtxt('tfidf.csv', delimiter=';')
```

Kode Sumber 4.16 Memuat File Berformat .csv ke Dalam Sebuah Array Numpy

```
3. from sklearn.model_selection import train_test_split
4. X = data_train[:, 1:]
5. y = data_train[:, 0].astype(np.int)
6. X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.3, random_state=0)
```

Kode Sumber 4.17 Membagi Data *Training* dan *Testing*

```
 7. from sklearn import svm
 8. clf_svm_linear = svm.NuSVC(kernel ='linear')
 9. clf_svm_linear = clf_svm_linear.fit(X_train, y_train)
10. clf_svm_rbf = svm.NuSVC(kernel = 'rbf')
11. clf_svm_rbf = clf_svm_rbf.fit(X_train,y_train)
12. clf_svm_poly = svm.NuSVC(kernel='poly', degree=3)
13. clf_svm_poly = clf_svm_poly.fit(X_train, y_train)
14. clf_svm_sigmoid = svm.NuSVC(kernel = 'sigmoid')
15. clf_svm_sigmoid = clf_svm_sigmoid.fit(X_train,y_train)
```

Kode Sumber 4.18 Melakukan *Training* dengan Menggunakan 4 *Kernel* yang Berbeda

Pada format file .csv untuk dataset ini, digunakan delimiter atau pembatas antar data yang berupa tanda baca, yaitu ‘;’ atau titik koma yang kemudian akan dimasukkan ke dalam *array* numpy seperti pada Kode Sumber 4.16 baris 1 dan 2. Selanjutnya, digunakan *library* dari scikit-learn untuk membagi antara dataset yang akan menjadi data *training* dan data *testing* seperti yang dilakukan pada Kode Sumber 4.17 baris 3 sampai 6.

Langkah berikutnya adalah menyertakan modul svm untuk menggunakan fungsi NuSVC yang merupakan fungsi untuk memanggil klasifier SVM yang menggunakan parameter *nu* dalam klasifikasinya seperti pada Kode Sumber 4.18 baris 7 sampai 15.

```
1. from sklearn.model_selection import cross_val_score
2. from sklearn.metrics import accuracy_score
```

Kode Sumber 4.19 Memanggil Library untuk Melakukan *Cross Validation* dan Mengeluarkan Akurasi dari Hasil Uji Coba

Untuk mengukur akurasi sistem, digunakan metode 10-*fold Cross Validation* menggunakan library dari sklearn seperti pada Kode Sumber 4.19 baris 1. Masing-masing fold menghasilkan akurasi yang berbeda-beda dan digunakan *library* dari *sklearn* seperti pada Kode Sumber 4.19 baris 2 untuk menghitung rata-rata dari semua fold.

```
1. hasil_test_svm_linear = clf_svm_linear.predict(X_test)
2. accuracy_svm_linear = accuracy_score(y_test,
   hasil_test_svm_linear)
3. print ("Kernel Linear = ")
4. print(accuracy_svm_linear)
5. scores_svm_1 = cross_val_score(clf_svm_linear, X, y,
   cv=10)
6. print(scores_svm_1)
7. print("\n")
```

Kode Sumber 4.20 Melakukan Prediksi SVM Menggunakan *Kernel* Linear

```
1. hasil_test_svm_rbf = clf_svm_rbf.predict(X_test)
2. accuracy_svm_rbf = accuracy_score(y_test,
   hasil_test_svm_rbf)
3. print ("Kernel RBF = ")
4. print(accuracy_svm_rbf)
5. scores_svm_2 = cross_val_score(clf_svm_rbf, X, y,
   cv=10)
6. print(scores_svm_2)
7. print("\n")
```

Kode Sumber 4.21 Melakukan Prediksi SVM Menggunakan *Kernel* RBF

```
1. hasil_test_svm_poly = clf_svm_poly.predict(X_test)
2. accuracy_svm_poly = accuracy_score(y_test,
   hasil_test_svm_poly)
3. print ("Kernel Polynomial = ")
4. print(accuracy_svm_poly)
5. scores_svm_3 = cross_val_score(clf_svm_poly, X, y,
   cv=10)
6. print(scores_svm_3)
7. print("\n")
```

Kode Sumber 4.22 Melakukan Prediksi SVM Menggunakan *Kernel* Polynomial

```
1. hasil_test_svm_sigmoid =
   clf_svm_sigmoid.predict(X_test)
2. accuracy_svm_sigmoid = accuracy_score(y_test,
   hasil_test_svm_sigmoid)
3. print ("Kernel Sigmoid = ")
4. print(accuracy_svm_sigmoid)
5. scores_svm_4 = cross_val_score(clf_svm_sigmoid, X, y,
   cv=10)
6. print(scores_svm_4)
7. print("\n")
```

Kode Sumber 4.23 Melakukan Prediksi SVM Menggunakan *Kernel* Sigmoid

Dalam mendapatkan hasil yang optimal, digunakan 4 *kernel* yang berbeda sehingga diharapkan ada *kernel* yang menghasilkan output dengan akurasi yang terbaik.

Kernel yang digunakan yaitu *Kernel* Linear pada Kode Sumber 4.20 baris 1 sampai 7, *Kernel* RBF pada Kode Sumber 4.21 baris 1 sampai 7, *Kernel* Polynomial seperti pada Kode Sumber 4.22 baris 1 sampai 7, dan *Kernel* Sigmoid seperti pada Kode Sumber 4.23 baris 1 sampai 7. Untuk *kernel* poly, *degree* yang merupakan *default* adalah 3, sedangkan untuk *kernel* yang lain, variabel *degree* ini diabaikan.

```
1. clf_nusvm_sigmoid_1 = svm.NuSVC(kernel = 'sigmoid',
   nu = 0.1)
2. clf_nusvm_sigmoid_1 =
   clf_nusvm_sigmoid_1.fit(X_train,y_train)
3. hasil_test_svm_sigmoid =
   clf_svm_sigmoid_1.predict(X_test)
```

Kode Sumber 4.24 Melakukan *Training* dan Prediksi Menggunakan Nilai Parameter *Nu* = 0,1

```
4.  AA, AB, AC, BA, BB, BC, CA, CB, CC = 0
5.  for x in range(0, 1095):
6.      if(y_test[x] == 0):
7.          if(hasil_test_nusvm_sigmoid[x]==y_test[x]):
8.              AA+=1
9.          elif(hasil_test_nusvm_sigmoid[x]==1):
10.             AB+=1
11.         else:
12.             AC+=1
13.     elif(y_test[x] == 1):
14.         if(hasil_test_nusvm_sigmoid[x]==0):
15.             BA+=1
16.         elif(hasil_test_nusvm_sigmoid[x]==y_test[x]):
17.             BB+=1
18.         else:
19.             BC+=1
20.     else:
21.         if(hasil_test_nusvm_sigmoid[x]==0):
22.             CA+=1
23.         elif(hasil_test_nusvm_sigmoid[x]==1):
24.             CB+=1
25.         else:
26.             CC+=1
```

Kode Sumber 4.25 Membuat Data untuk Matriks

4.7. Implementasi Modul Pembuatan Matriks

Pada modul sebelumnya telah didapatkan hasil akurasi dan nilai dari 10-*fold cross validation*. Dari hasil tersebut, dapat terlihat akurasi dari masing-masing *kernel* dan akan diambil satu *kernel* dengan akurasi paling tinggi yaitu *Kernel* Sigmoid untuk percobaan selanjutnya yaitu dengan mengubah nilai parameter *nu* seperti pada Kode Sumber 4.24 baris 1 sampai 3 yang meggunakan nilai parameter *nu* sebesar 0,1.

Selanjutnya, hasil yang didapatkan dapat dibentuk menjadi sebuah matriks berukuran n x n di mana n adalah jumlah kelas. Matriks ini akan berisi jumlah *tweet* yang berhasil diklasifikasikan sebagai label yang sebenarnya maupun jumlah *tweet* yang mengalami penyimpangan hasil saat klasifikasi.

Untuk membuat matriks, diperlukan variabel untuk menampung masing-masing hasil klasifikasi per kelas berjumlah 9 variabel seperti pada Kode Sumber 4.25 baris 4.

Dikarenakan data yang menjadi prediksi berjumlah 1095, maka diperlukan perulangan sebanyak 1095 kali untuk mengecek hasil klasifikasi label setiap datanya seperti pada Kode Sumber 4.25 baris 4 sampai 26.

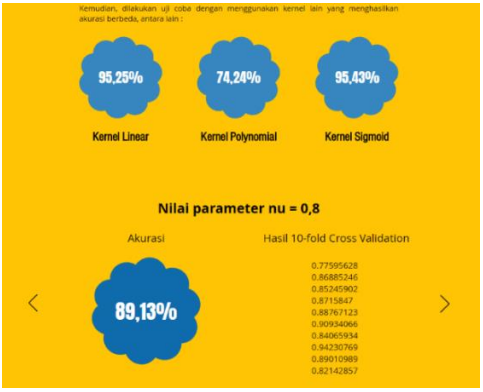
4.8. Implementasi Pemuatan Hasil ke Dalam Halaman *Web*

Terakhir adalah memuat hasil yang telah didapatkan dari klasifikasi berupa akurasi sistem yang menggunakan klasifier NuSVC dengan *kernel* dan nilai parameter *nu* awal yaitu *Kernel* RBF dan *Nu* 0,5 yang menghasilkan akurasi seperti yang dapat dilihat pada Gambar 4.1.

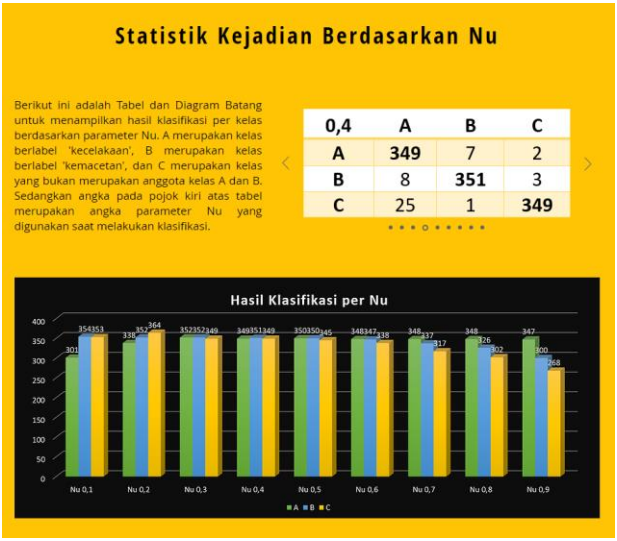
Kemudian, akan ditampilkan akurasi dari 3 *kernel* lainnya seperti pada Gambar 4.2 sekaligus juga ditampilkan hasil dari penghitungan 10-fold Cross Validation yang digunakan untuk menghitung akurasi sistem.

Gambar 4.1 Halaman *Web* yang Menampilkan Akurasi Awal Sistem

Kemudian, akan ditampilkan akurasi dari 3 *kernel* lainnya seperti pada Gambar 4.2 sekaligus juga ditampilkan hasil dari penghitungan 10-fold Cross Validation yang digunakan untuk menghitung akurasi sistem. Terakhir, akan ditampilkan juga statistik dari jumlag kejadian yang berhasil diklasifikasikan dengan tepat menggunakan diagram batang yang dapat dilihat pada Gambar 4.3.



Gambar 4.2 Halaman *Web* yang Menampilkan Akurasi Sistem dan Hasil Cross Validation



Gambar 4.3 Hasil Statistik Kejadian

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini, dibahas uji coba dan evaluasi implementasi metode-metode yang digunakan. Hal-hal yang diujikan dalam bab ini adalah: akurasi hasil klasifikasi dari metode SVM untuk mengklasifikasikan *tweet* yang masuk kedalam dataset dengan membandingkan performa dari 4 *kernel* dan relasi antara perubahan nilai parameter *nu* dengan akurasi hasil klasifikasi SVM.

5.1. Lingkungan Uji Coba

Lingkungan uji coba dalam tugas akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk implementasi metode. Perangkat keras yang digunakan adalah computer dengan prosesor Intel® Core™ i7-7700HQ dengan kecepatan 2.80 GHz dan memori 8 gigabyte RAM. Uji coba dilakukan di sistem operasi Microsoft Windows 10 64bit dengan kakas bantu Python 3.6.1.

5.2. Data Uji Coba

Data yang digunakan untuk uji coba adalah data *tweet* yang didapatkan melalui akun-akun yang telah ditentukan pada batasan masalah menggunakan REST API Twitter maupun *tweet* yang mengandung kata kecelakaan dan kemacetan yang diambil melalui *Search* API Twitter.

Data *tweet* yang didapatkan memiliki atribut yaitu teks/isi dari *tweet* dan waktu kapan *tweet* dibuat. Data yang digunakan dalam tugas akhir ini adalah data *tweet* dari tanggal 1 Januari 2017 sampai dengan 1 November 2017. *Tweet* yang telah diambil, akan dipreproses dan diberikan label yaitu Kecelakaan, Kemacetan, atau Lain-Lain yang contohnya seperti pada Tabel 5.1. Di dalam databasenya Kecelakaan diwakili dengan angka 1, Kemacetan dengan angka 2, dan Lain-Lain dengan angka 0.

Tabel 5.1 Contoh Tweet, Hasil Preproses, dan Labelnya

Tweet	Hasil Preproses	Label
RT @rezadjwika: @e100ss terjadi kemacetan turunan trosobo menuju surabaya, karena kecelakaan. Sepertinya sepeda masuk kolong treller	jadi macet turun trosobo tuju surabaya **celaka** sepeda masuk kolong treller	Kecelakaan
RT @wahyuniswi: @e100ss tol dupak-satelit padat sejak gerbang tol dominasi kendaraan besar	tol dupak-satelit **padat** sejak gerbang tol dominasi kendara besar	Kemacetan
RT @muchsoli: @e100ss Arus lalin simpang 4 secang magelang ramai lancar cuaca cerah berawan	arus lalin simpang 4 secang magelang ramai lancar cuaca cerah awan	Lain-Lain

Pada tweet yang dikategorikan sebagai anggota Kelas Kecelakaan pada Tabel 5.1, terdapat kata macet di dalamnya. Namun, dikarenakan pada *labeling* manual terdapat hirarki atau urutan dalam pembuatan kondisinya, yaitu pada kondisi pertama jika dalam sebuah *tweet* terdapat kata yang mengindikasikan terjadinya kecelakaan, maka *tweet* akan langsung dianggap sebagai anggota dari Kelas Kecelakaan.

Adapun urutan kondisinya yaitu Kecelakaan terlebih dahulu, diikuti dengan Kemacetan, kemudian terakhir yaitu Lain-Lain. Hal ini dipertimbangkan dari peristiwa aktual di kejadian nyatanya yang biasanya memang kecelakaan secara tidak langsung dapat menyebabkan kemacetan. Sehingga, untuk sistem yang dibuat dalam tugas akhir ini, kecelakaan didahulukan.

Dari semua dokumen atau semua *tweet*, ditemukan sebanyak 39 tweet yang mengandung kata-kata yang dapat mengategorikan tweet ke dalam dua kelas sekaligus, kemacetan maupun kecelakaan. Namun, dikarenakan kondisi hirarki yang telah dibuat pada saat implementasi adalah mendahulukan kata-kata yang berhubungan dengan kecelakaan, maka 39 tweet ini dimasukkan sebagai anggota Kelas Kecelakaan.

5.3. Skenario Uji Coba

Di bagian ini dijelaskan skenario uji coba yang dilakukan. Ada 2 skenario uji coba, yaitu uji coba akurasi metode *Support Vector Machine* dengan menggunakan 4 *kernel* yang berbeda, dan uji coba perubahan nilai parameter *nu* terhadap akurasi klasifikasi metode *Support Vector Machine*.

5.3.1. Skenario Uji Coba Performa *Kernel* pada Metode SVM

Pada skenario ini, digunakan 3649 *tweet* sebagai data *training* dan diambil 30% dari data, yaitu 1095 data untuk data *testing*. Data yang diambil merupakan *tweet* yang memenuhi kriteria pada bab 1.3 dari tanggal 12 September 2017 sampai 16 Desember 2017. Dilakukan klasifikasi secara manual pada tulisan di *tweet*, apakah *tweet* mengandung informasi tentang kecelakaan atau kemacetan atau lain-lain.

Setelah dilakukan *preprocessing*, dilakukan ekstraksi fitur pada *tweet*. Fitur yang diambil merupakan representasi dari TF-IDF milik semua *tweet* yang ada di dalam dataset.

Masing-masing fitur akan dievaluasi dengan metode SVM sebanyak 4 kali dengan *kernel* yang berbeda-beda, yaitu *kernel* Linear, RBF, Poly, dan Sigmoid. Untuk proses evaluasi akan dilakukan dengan 10-fold *cross-validation*.

5.3.2. Skenario Uji Coba Akurasi Klasifikasi SVM dengan Perubahan Nilai Parameter *Nu*

Pada skenario ini digunakan dataset yang sama dengan sebelumnya dan semua *tweet* diklasifikasikan menggunakan metode SVM menggunakan *kernel* terbaik yang didapat dari uji coba sebelumnya sebanyak 9 kali dengan mengubah nilai parameter *nu* mulai dari 0,1 sampai 0,9 untuk melihat besar penalti akibat kesalahan dalam klasifikasi data, yaitu semakin besar nilai parameter regularisasinya, maka semakin besar penalti yang diberikan sehingga akurasi klasifikasinya menurun.

5.4. Hasil Uji Coba

Pada bagian ini akan dijelaskan hasil evaluasi yang telah didapatkan dari klasifikasi menggunakan SVM.

5.4.1. Hasil Uji Coba Performa *Kernel* pada Metode SVM

Hasil uji coba metode klasifikasi *tweet* dengan SVM dilakukan dengan metode evaluasi 10-fold *cross-validation*. Hasil dari evaluasi ditunjukkan pada Tabel 5.2.

Tabel 5.2 Hasil Uji Coba Akurasi Metode Klasifikasi

Nama *Kernel*	**Akurasi (%)**
Linear	95,25
Radial Basis Function	95,15
Polynomial	74,24
Sigmoid	**95,43**

Dari data pada Tabel 5.2 terlihat bahwa metode SVM dengan menggunakan *kernel* default yaitu RBF memiliki performa yang baik. Namun, bila dibandingkan dengan 3 *kernel* lainnya, maka akurasi yang paling tinggi merupakan hasil akurasi klasifikasi menggunakan ***Kernel* Sigmoid**. Oleh karena itu *kernel* yang memiliki performa paling baik pada dataset ini adalah *Kernel* Sigmoid.

5.4.2. Hasil Uji Coba Akurasi Klasifikasi SVM dengan Perubahan Nilai Parameter *Nu*

Berikut adalah hasil akurasi klasifikasi SVM menggunakan *Kernel* Sigmoid dengan masing-masing nilai *Nu* yang digunakan beserta dengan hasil 10-fold cross validation dapat dilihat pada Tabel 5.2.

Dari hasil akurasi per nilai *nu* pada Tabel 5.3, dapat dibuat sebuah grafik untuk melihat hasil performa dari nilai *nu* seperti yang dapat dilihat pada Gambar 5.1. Tampak bahwa kecenderungan relasi antara nilai parameter *nu* dengan akurasi adalah semakin besar nilai parameter *nu*, maka semakin kecil akurasi yang didapatkan.

Tabel 5.3 Daftar Akurasi dengan Mengubah *Nu*

Nilai Parameter *Nu*	Akurasi (%)
0,1	92,05
0,2	**96,25**
0,3	96,16
0,4	95,79
0,5	95,43
0,6	94,33
0,7	91,50
0,8	89,13
0,9	83,56

Oleh karena itu, hasil akurasi yang paling baik didapatkan pada saat nilai parameter *nu* adalah sebesar **0,2** dengan akurasi sebesar **96,25%.**

5.5. Analisis Hasil Uji Coba

Selain akurasi, dibuat juga sebuah matriks yang berukuran 3x3 untuk menganalisis pola bias yang muncul pada saat klasifikasi. Dapat dilihat pada Gambar 5.2 yang menunjukkan hasil

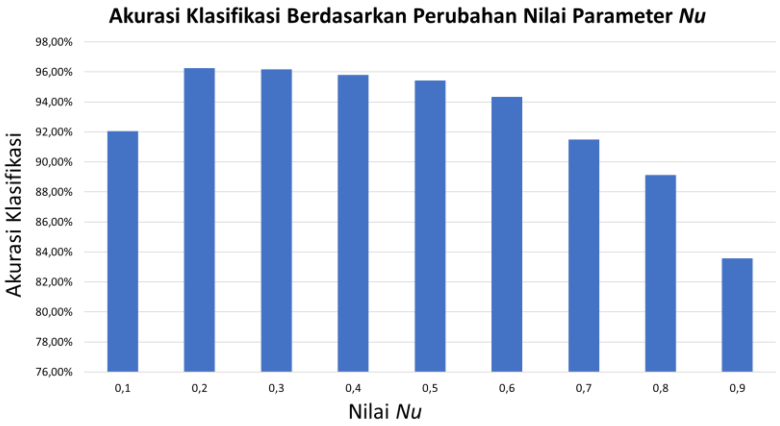
klasifikasi menggunakan SVM dengan nilai *nu* sebesar 0,2. Kolom menunjukkan kelas hasil klasifikasi data sementara baris menunjukkan kelas yang sebenarnya. Huruf A merupakan Kelas Kecelakaan, huruf B merupakan Kelas Kemacetan, sedangkan huruf C merupakan Kelas Lain-Lain. Terlihat bahwa *tweet* yang seharusnya berada di dalam kelas A, ternyata sebanyak 17 *tweet* diklasifikasikan sebagai kelas C. Begitu pula dengan kelas C, sebanyak 11 *tweet* diklasifikasikan sebagai kelas A.

Hal ini membuktikan bahwa kelas A dan C memiliki banyak fitur yang mirip dibandingkan dengan kelas B, bahkan tidak ada 1 *tweet* pun yang diklasifikasikan sebagai kelas B pada kelas C yang sebenarnya. Sementara itu, pada kelas B yang sebenarnya dapat dilihat 8 *tweet* diklasifikasikan sebagai anggota kelas A sedangkan 2 *tweet* diklasifikasikan sebagai anggota kelas C.

Selanjutnya pada Gambar 5.3, dapat dilihat confusion matrix untuk nilai *nu* sebesar 0,9. Terlihat bahwa kelas A yang sebenarnya memiliki 6 anggota yang diklasifikasikan sebagai anggota kelas C yang berarti jumlahnya lebih banyak dari anggota yang diklasifikasikan sebagai anggota kelas B yaitu sebanyak 5 *tweet*, namun biasnya tidak terlalu besar. Perilaku ini serupa dengan pada saat menggunakan nilai *nu* sebesar 0,2.

Di sisi lain, anggota kelas C yang diklasifikasikan sebagai kelas A mencapai angka 103 yang mana hampir merupakan separuh dari banyaknya anggota kelas C yang diklasifikasikan dengan tepat yaitu 268 *tweet*. Kemudian, untuk anggota kelas B, sama seperti sebelumnya yaitu ada lebih banyak *tweet* yang dikategorikan sebagai kelas A daripada kelas C.

Maka dapat ditarik kesimpulan bahwa anggota kelas A tidak terlalu memiliki kemiripan dengan kelas yang lain berdasarkan fiturnya, anggota kelas B lebih mirip dengan kelas A dibandingkan dengan kelas C, sedangkan anggota kelas C memiliki kemiripan yang dekat dengan anggota kelas A.



Gambar 5.1 Grafik Perubahan Akurasi SVM Terhadap Nilai Parameter *Nu*

Nu : 0,2		Prediksi A	Prediksi B	Prediksi C
Aktual	A	**338**	3	17
	B	8	**352**	2
	C	11	0	**364**

Gambar 5.2 *Confusion Matrix* pada Skenario Terbaik *Nu* = 0,2

Nu : 0,9		Prediksi A	Prediksi B	Prediksi C
Aktual	A	**347**	5	6
	B	53	**300**	9
	C	103	4	**268**

Gambar 5.3 *Confusion Matrix* pada Skenario Terburuk *Nu* = 0,9

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Dalam melakukan klasifikasi *tweet* ke dalam tiga kelas, metode SVM memiliki akurasi yang paling baik yaitu 96,25% apabila menggunakan *Kernel* Sigmoid dan parameter *nu* sebesar 0,2.
2. Nilai akurasi yang didapatkan melalui uji coba performa *kernel* mirip antara satu sama lain yaitu di atas 95% kecuali untuk *Kernel* Polynomial yang menghasilkan akurasi hanya 74%.
3. Kecenderungan perubahan parameter *nu* terhadap hasil akurasi klasifikasi dengan metode SVM adalah semakin besar nilai parameter *nu*, maka semakin kecil hasil akurasi yang didapat.
4. Pada Tugas Akhir ini, berdasarkan hasil analisis dari *Confusion Matrix*, anggota 'Kelas Kecelakaan' lebih mirip dengan anggota 'Kelas Lain-Lain' daripada 'Kelas Kemacetan', anggota 'Kelas Kemacetan' lebih mirip dengan 'Kelas Kecelakaan' dibandingkan dengan kelas lain-lain, sedangkan anggota 'Kelas Lain-Lain' memiliki kemiripan yang dekat dengan anggota 'Kelas Kecelakaan'.

6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Data yang didapatkan sekarang adalah data sampling yang diberikan oleh Twitter. Untuk mendapatkan semua *tweet* yang masuk ke Twitter, dibutuhkan akses ke Firehose API yang membutuhkan biaya. Kedepannya hal ini bisa dikembangkan lebih lanjut.
2. Adanya metadata dari *tweet* yang memuat tentang lokasi dari kejadian sehingga aplikasi dapat dikembangkan untuk disambungkan dengan Google Maps API.
3. Dalam tugas klasifikasi, dapat diimplementasikan metode klasifikasi lain seperti *Random Forest* atau menggunakan metode turunan SVM lainnya seperti *Support Vector Regression.*
4. Pada saat menggunakan *Kernel* Polynomial, dapat dilakukan uji coba berbagai skenario berdasarkan perubahan variabel *degree* dan *C* untuk mencari hasil yang optimal dari *Kernel* Polynomial.

DAFTAR PUSTAKA

[1] Twitter, "About Company," 2015. [Online]. Available: https://about.twitter.com/company.

[2] Wikipedia, "Application Programming Interface," 2015. [Online]. Available: http://en.wikipedia.org/wiki/Application_programming_interface.

[3] Twitter, "Twitter API Documentation," 2015. [Online]. Available: https://dev.twitter.com/overview/documentation.

[4] V. Jakkula, "Tutorial on Support Vector Machine," 2013.

[5] E. Susilowati, M. K. Sabariah and A. A. Gozali, "IMPLEMENTATION SUPPORT VECTOR MACHINE METHOD FOR TRAFFIC JAM," 2015.

[6] R. Diani, "Analisis Pengaruh *Kernel* Support Vector Machine (SVM) pada Klasifikasi Data Microarray untuk Deteksi Kanker," *Indonesian Journal of Computing*, vol. 2, no. 1, pp. 109-118, 2017.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Tabel 6.1 Hasil Uji Coba Akurasi Metode Klasifikasi SVM menggunakan Kernel Sigmoid dengan 10-Fold *Cross Validation*

Nilai *Nu*	Fold ke	Hasil Cross Validation
0,1	1	0.87431694
	2	0.94808743
	3	0.93715847
	4	0.95628415
	5	0.96164384
	6	0.95054945
	7	0.93131868
	8	0.96153846
	9	0.92032967
	10	0.94230769
0,2	1	0.92076503
	2	0.96448087
	3	0.95901639
	4	0.98360656
	5	0.97534247
	6	0.96428571
	7	0.9532967
	8	0.98076923
	9	0.97252747
	10	0.96428571
0,3	1	0.91803279
	2	0.95901639
	3	0.96448087
	4	0.9726776
	5	0.96712329
	6	0.97252747
	7	0.95879121

	8	0.98626374
	9	0.98351648
	10	0.93681319
0,4	1	0.90163934
	2	0.9726776
	3	0.95355191
	4	0.9726776
	5	0.96164384
	6	0.97802198
	7	0.93131868
	8	0.98351648
	9	0.96703297
	10	0.93681319
0,5	1	0.89344262
	2	0.95628415
	3	0.90983607
	4	0.93715847
	5	0.94794521
	6	0.95054945
	7	0.92307692
	8	0.98076923
	9	0.96153846
	10	0.94230769
0,6	1	0.86612022
	2	0.94262295
	3	0.89344262
	4	0.91530055
	5	0.93150685
	6	0.92307692
	7	0.89010989
	8	0.97527473
	9	0.94505495
	10	0.9010989
0,7	1	0.81967213

	2	0.91530055
	3	0.86885246
	4	0.8852459
	5	0.91506849
	6	0.91208791
	7	0.87637363
	8	0.96703297
	9	0.93131868
	10	0.88461538
0,8	1	0.77595628
	2	0.86885246
	3	0.85245902
	4	0.8715847
	5	0.88767123
	6	0.90934066
	7	0.84065934
	8	0.94230769
	9	0.89010989
	10	0.82142857
0,9	1	0.73497268
	2	0.84153005
	3	0.80054645
	4	0.83879781
	5	0.86027397
	6	0.8956044
	7	0.81593407
	8	0.9010989
	9	0.81593407
	10	0.78296703

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis, **Vessa Rizky Oktavia**, lahir di Surabaya pada 5 Oktober 1995 yang merupakan Hari ABRI. Penulis adalah anak pertama dari dua bersaudara dan memiliki adik kandung laki-laki dengan selisih usia 4 tahun. Penulis menempuh pendidikan formal di SDN Rangkah VI Surabaya (2000-2006), SMPN 9 Surabaya (2006-2009), dan SMAN 5 Surabaya (2009-2012). Mulai tahun 2012, penulis menempuh pendidikan S1 di Departemen Informatika, FTIK, Institut Teknologi Sepuluh Nopember Surabaya.

Di Departemen Informatika, penulis mengambil bidang minat KCV dan memiliki minat pada beberapa subjek seperti *machine learning* dan *data mining*. Penulis sempat aktif di Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi dan Komunikasi tahun 2013 sebagai Staff Divisi Hubungan Luar, kemudian menjadi Staff Ahli Divisi Pengembangan Sumber Daya Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika ITS pada tahun 2014 dan menjadi Panitia *Schematics* 2013 dan 2014 sebagai Anggota Divisi *National Logic Competition*. Penulis senang melakukan kegiatan yang berhubungan dengan *eSports* dan *game*. Penulis dapat dihubungi melalui alamat email vessango9@gmail.com