

**KLASIFIKASI TEKS BERBAHASA INDONESIA PADA
TWITTER MENGGUNAKAN METODE *BACKPROPAGATION*
ARTIFICIAL NEURAL NETWORK PADA BADAN NASIONAL
PENANGGULANGAN BENCANA (BNPB)**

TUGAS AKHIR

Oleh:

HELMI NUR IHSAN

NIM: 1511501551

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

JAKARTA

2019

**KLASIFIKASI TEKS BERBAHASA INDONESIA PADA
TWITTER MENGGUNAKAN METODE *BACKPROPAGATION*
ARTIFICIAL NEURAL NETWORK PADA BADAN NASIONAL
PENANGGULANGAN BENCANA (BNPB)**

**Diajukan untuk memenuhi salah satu persyaratan memperoleh
gelar Sarjana Komputer (S.Kom)**

TUGAS AKHIR

Oleh:

HELMI NUR IHSAN

NIM: 1511501551

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR**

JAKARTA

2019



**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR**

LEMBAR PENGESAHAN

Nama : Helmi Nur Ihsan
Nomor Induk Mahasiswa : 1511501551
Program Studi : Teknik Informatika
Bidang Peminatan : *Programming Expert*
Jenjang Studi : Strata 1
Judul : Klasifikasi Teks Berbahasa Indonesia pada Twitter
Menggunakan Metode Backpropagation Artificial
Neural Network pada Badan Nasional
Penanggulangan Bencana (BNPB)

Jakarta, 2 Agustus 2019

Tim Penguji :

Ketua,
Sri Mulyati, S.Kom., M.Kom

Anggota,
Subandi, S.Kom., M.Kom

Pembimbing,
Dr. Indra, S.Kom., M.T.I

Tanda Tangan :

Ketua Program Studi

Dr. Indra, S.Kom., M.T.I

ABSTRAK

KLASIFIKASI TEKS BERBAHASA INDONESIA PADA TWITTER MENGGUNAKAN METODE BACKPROPAGATION ARTIFICIAL NEURAL NETWORK PADA BADAN NASIONAL PENANGGULANGAN BENCANA (BNPB)

Oleh: Helmi Nur Ihsan (1511501551)

Media sosial telah mengalami perkembangan yang sangat pesat, dengan jumlah penggunanya yang sangat masif, membuat penyebaran informasi menjadi sangat luas dan cepat. Selain memudahkan pengguna untuk saling berinteraksi dan bersosialisasi, media sosial juga dapat dimanfaatkan untuk mendapatkan informasi mengenai bencana alam yang terjadi, karena kecendrungan pengguna yang akan segera memberikan *update* keadaan terkini seputar bencana alam yang dialami. Twitter sebagai salah satu penyedia jasa media sosial, tercatat memiliki pengguna yang berasal dari Indonesia sebesar 19,5 juta orang. Dengan tersedianya sumber informasi tersebut, akan terasa sulit jika harus memeriksa informasi yang dibagikan oleh pengguna satu per satu, dan mengkalsifikasi apakah informasi tersebut benar memiliki subjek bencana longsor atau bukan. Oleh karena itu, diperlukan suatu implementasi algoritme yang dapat mengatasi masalah tersebut. Untuk menyusun tugas akhir ini penulis mengambil judul “Klasifikasi Teks Berbahasa Indonesia pada Twitter Menggunakan Metode *Backpropagation Artificial Neural Network* pada Badan Nasional Penanggulangan Bencana (BNPB)”. Metode ini memiliki kelebihan mampu melakukan klasifikasi secara akurat dengan memberikan nilai bobot pada setiap kata yang terdapat dalam informasi yang didapat dari Twitter, sehingga hasil klasifikasi lebih akurat dan tidak bias. Dengan hasil pengujian yang memberikan nilai akurasi sebesar 98% dan tingkat nilai kesalahan hanya sebesar 0,01756.

Kata kunci: *Backpropagation, Text Classification, Media Sosial, Bencana Longsor*

SURAT PERNYATAAN TIDAK PLAGIAT DAN PERSETUJUAN PUBLIKASI

Saya yang bertanda tangan dibawah ini :

Nama : Helmi Nur Ihsan
Nomor Induk Mahasiswa : 1511501551
Program Studi : Teknik Informatika
Bidang Peminatan : *Programming Expert*
Jenjang Studi : Strata 1
Fakultas : Teknologi Informasi

menyatakan bahwa TUGAS AKHIR yang berjudul:

KLASIFIKASI TEKS BERBAHASA INDONESIA
PADA TWITTER MENGGUNAKAN METODE
BACKPROPAGATION ARTIFICIAL NEURAL
NETWORK PADA BADAN NASIONAL
PENANGGULANGAN BENCANA (BNPB)

Merupakan :

1. Karya tulis saya sebagai laporan tugas akhir yang asli dan belum pernah diajukan untuk mendapatkan gelar akademik apapun, baik di Universitas Budi Luhur maupun di perguruan tinggi lainnya.
2. Karya tulis ini bukan saduran / terjemahan, dan murni gagasan, rumusan dan pelaksanaan penelitian / implementasi saya sendiri, tanpa bantuan pihak lain, kecuali arahan pembimbing akademik dan pembimbing di organisasi tempat riset.
3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Saya menyerahkan hak milik atas karya tulis ini kepada Universitas Budi Luhur, dan oleh karenanya Universitas Budi Luhur berhak melakukan pengelolaan atas karya tulis ini sesuai dengan norma hukum dan etika yang berlaku.

Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh berdasarkan karya tulis ini, serta sanksi lainnya sesuai dengan norma di Universitas Budi Luhur dan Undang-Undang yang berlaku

Jakarta, 30 Juli 2019



Helmi Nur Ihsan

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT penulis panjatkan atas segala rahmat dan hidayah-Nya sehingga penulis bisa menyelesaikan Laporan Tugas Akhir berjudul “Klasifikasi Teks Berbahasa Indonesia pada Twitter Menggunakan Metode Backpropagation Artificial Neural Network pada Badan Nasional Penanggulangan Bencana (BNPB)”. Adapun, diajukannya laporan tugas akhir ini bertujuan untuk memenuhi syarat guna menyelesaikan tugas akhir. Laporan ini disusun berdasarkan data yang sudah dikumpulkan. Diajukannya laporan tugas akhir ini untuk memenuhi salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) di Fakultas Teknologi Informasi Universitas Budi Luhur.

Dalam proses penyelesaian laporan tugas akhir ini, penulis menyampaikan terima kasih sebesar-besarnya kepada berbagai pihak yang sudah memberikan dukungan dan bantuan, terutama kepada:

1. Allah SWT, Tuhan Yang Maha Esa yang selalu memberikan rahmat, nikmat, dan hidayah-Nya sehingga penulis bisa menyelesaikan penyusunan tugas akhir ini dengan baik.
2. Segenap keluarga penulis, terutama kepada orang tua penulis karena telah membantu penulis dengan memberikan dukungan, baik moral maupun material, dan selalu memberikan doa, restu, perhatian, serta kasih sayang kepada penulis.
3. Rektor Universitas Budi Luhur Jakarta, Bapak Dr. Ir. Wendi Usino, M.M., M.sc
4. Bapak Dr. Deni Mahdiana, S.Kom., M.M., M.Kom selaku Dekan Fakultas Teknologi Informasi Universitas Budi Luhur.
5. Bapak Dr. Indra, S.Kom, M.T.I selaku Kaprodi Teknik Informatika Fakultas Teknologi Informasi Universitas Budi Luhur dan selaku Dosen Pembimbing yang sudah membantu penulis dan memberikan saran dalam penulisan laporan tugas akhir ini.
6. Seluruh dosen Universitas Budi Luhur yang sudah memberikan ilmu yang bermanfaat bagi penulis.

7. Badryatusyahryah selaku *support system* yang tidak pernah bosan memberikan dukungan dan semangat untuk penulis.
8. Para asisten dan supervisor Lab ICT Terpadu Universitas Budi Luhur yang sudah menyediakan sarana pembelajaran bagi penulis selama menuntut ilmu di Universitas Budi Luhur.
9. Teman-teman dekat penulis, yaitu Bang Ade, Adam Abimanyu, Alfath Dioni, Samuel Niko, Setyo Handoko, Fauzi Setyo Fendhio, Edi Susanto, Agung Santoso, dan Pandhu Wibowo.
10. Semua pihak yang terkait, baik secara langsung maupun tak langsung dalam penulisan laporan tugas akhir ini dari tahap awal sampai tahap akhir.

Semoga Allah SWT selalu melimpahkan karunia dan anugerah-Nya atas segala bantuan yang sudah diberikan. Aamiin.

Dalam penulisan tugas akhir ini, penulis menyadari bahwa masih banyak kekurangan, kelemahan, serta kesalahan yang terdapat dalam penulisan tugas akhir ini. Oleh karena itu, kritik dan saran yang membangun dari berbagai pihak sangat diharapkan sebagai masukan untuk bahan evaluasi penulis. Akhir kata, penulis memohon maaf atas segala kekurangan yang terdapat dalam tugas akhir ini. Semoga tugas akhir ini berguna untuk semua pihak.

Jakarta, Agustus 2019

Penulis

DAFTAR TABLE

Tabel 3.1 Dataset	23
Tabel 3.2 Daftar <i>Term</i> pada <i>Input Layer</i>	38
Tabel 3.3 Daftar Bobot Sinapsis 1 (<i>w</i>).....	40
Tabel 3.4 Daftar Bobot Sinapsis 2 (<i>wh</i>).....	41
Tabel 3.5 Perhitungan <i>Feed Forward</i> pada <i>Hidden Layer</i>	42
Tabel 3.6 Perhitungan <i>Feed Forward</i> pada <i>Output Layer</i>	42
Tabel 3.7 Nilai <i>Error</i> pada <i>Output Layer</i>	43
Tabel 3.8 Nilai <i>Delta</i> (Perubahan) pada <i>Output Layer</i>	43
Tabel 3.9 Nilai <i>Error</i> pada <i>Hidden Layer</i>	43
Tabel 3.10 Nilai <i>Delta</i> (Perubahan) pada <i>Hidden Layer</i>	44
Tabel 3.11 Nilai Pembaruan Sinapsis 2	44
Tabel 3.12 Nilai Pembaruan Sinapsis 1	45
Tabel 3.13 Nilai Bobot Baru pada <i>Input Layer</i> (Sinapsis 1)	45
Tabel 3.14 Nilai Bobot Baru pada <i>Hidden Layer</i> (Sinapsis 2)	46
Tabel 4.1 Total Square Error Seluruh Data Testing	58
Tabel 4.2 Perhitungan Tingkat Akurasi Algoritme <i>Backpropagation</i>	60



DAFTAR GAMBAR

Gambar 2.1 Struktur Arsitektur MLP Jaringan Saraf Tiruan	7
Gambar 2.2 Diagram proses <i>Neural Network</i>	8
Gambar 2.3 Diagram pengaturan bobot pada <i>Supervised Learning</i>	9
Gambar 2.4 Arsitektur Jaringan <i>Backpropagation</i> (Riswanto dkk., 2014)	11
Gambar 3.1 <i>Flowchart</i> Proses Pengumpulan Data <i>Scraping</i>	25
Gambar 3.2 <i>Flowchart Preprocessing</i>	26
Gambar 3.3 <i>Flowchart</i> Seleksi <i>Input Layer</i>	28
Gambar 3.4 <i>Flowchart</i> Algoritme <i>Backpropagation</i> Bag. 1	30
Gambar 3.5 Flowchart Algoritme <i>Backpropagation</i> Bag. 2	31
Gambar 3.6 <i>Flowchart</i> Algoritme Prediksi <i>Feed Forward</i>	33
Gambar 3.7 <i>Flowchart</i> Menu <i>Monitoring</i> dan <i>Live Streaming tweet</i>	34
Gambar 3.8 <i>Flowchart</i> Menu <i>Update</i> Bobot Sinapsis.....	36
Gambar 3.9 Ilustrasi Arsitektur yang Digunakan	39
Gambar 3.10 Ilustrasi Bobot pada <i>Neural Network</i>	40
Gambar 3.11 Rancangan Layar <i>List Dataset</i>	47
Gambar 3.12 Rancangan Layar Halaman Prediksi	48
Gambar 3.13 Rancangan Layar Halaman Prediksi Spesifik Data	48
Gambar 4.1 Akses Halaman Twitter Advanced Search	50
Gambar 4.2 Isi Kriteria Pencarian	50
Gambar 4.3 Hasil Pencarian Twitter Advanced Search	51
Gambar 4.4 Konfigurasi <i>Sitemap</i> Web Scraper.....	51
Gambar 4.5 Hasil <i>Scraping Data</i>	52
Gambar 4.6 Optimasi Format Data Pada Excel dan Pelabelan Manual	52
Gambar 4.7 Proses <i>Training Data</i>	53
Gambar 4.8 Hasil <i>Training Data</i>	53
Gambar 4.9 Tampilan Layar Utama	55
Gambar 4.10 Tampilan Halaman Prediksi.....	56
Gambar 4.11 Halaman Prediksi Data Pilihan	57

DAFTAR SIMBOL

1. Simbol Activity Diagram

Simbol	Deskripsi
	Initial Node Awal aktifitas.
	Final Node Akhir aktifitas.
	Action Node Menggambarkan proses atau kegiatan bisnis.
	Decision Node Menggambarkan pilihan untuk pengambilan keputusan, true atau false.
	Partition Yaitu pembagian aktifitas untuk menunjukkan siapa melakukan apa.
	Association Menggambarkan hubungan antara <i>initial node</i> , <i>action node</i> , <i>decision point</i> dan <i>activity final node</i> dalam sebuah sistem.

2. Simbol Class Diagram

Simbol	Deskripsi
	Atribut Properti dari sebuah class. atribut melambangkan batas nilai yang mungkin ada pada objek dari class.
	Name Menggambarkan nama dari <i>class</i> atau objek
	Operation Menggambarkan implementasi dari layanan yang dapat diminta dari beberapa object dari <i>class</i> , yang mempengaruhi behavior
	Association Sebuah asosiasi merupakan sebuah <i>relationship</i> paling umum antara 2 <i>class</i> dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 <i>class</i> .

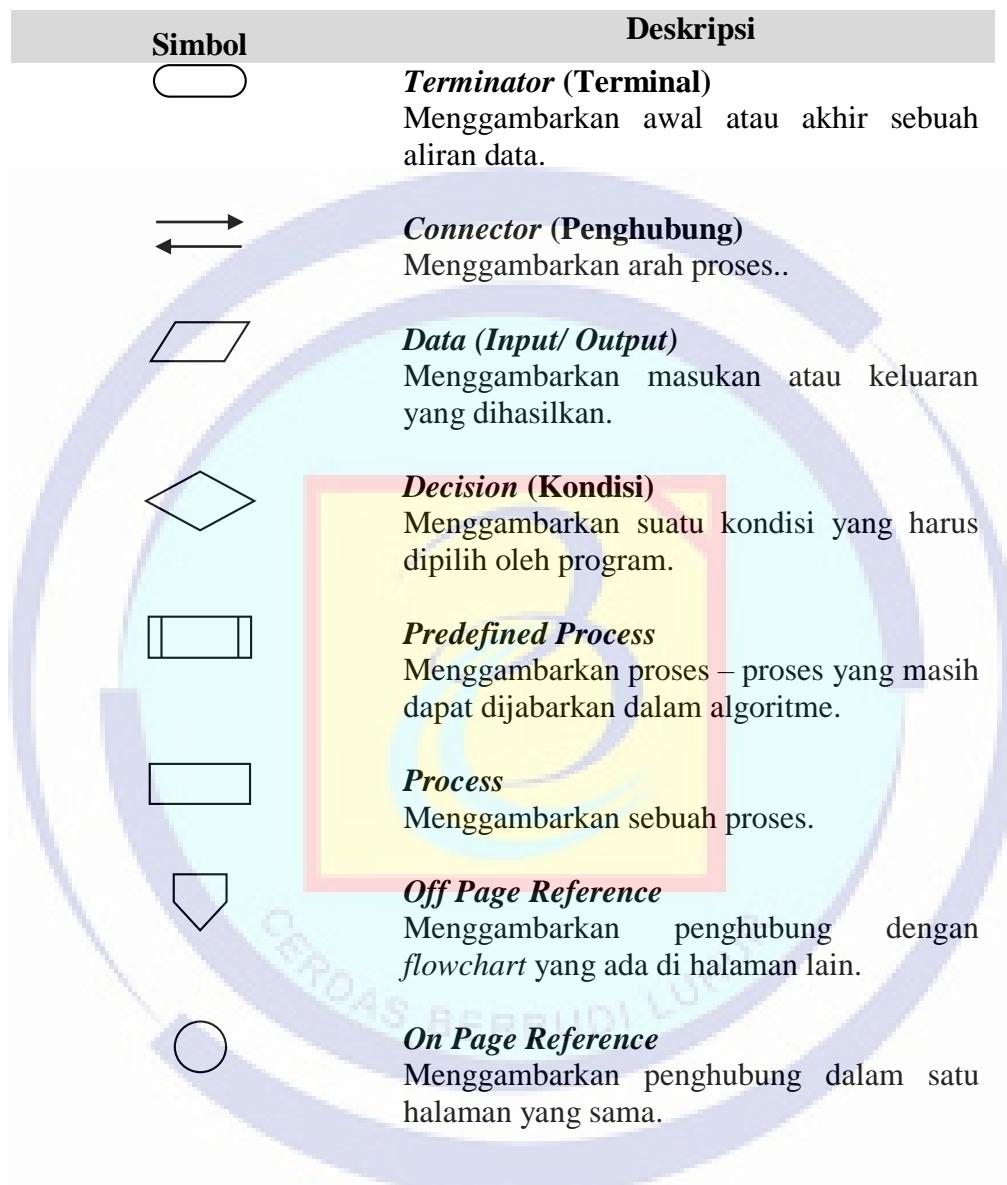
3. Simbol *Logical Record Structure*

Simbol	Deskripsi
	Entity (Entitas) Entity adalah sesuatu apa saja yang ada didalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data.
	Relationship Relationship adalah hubungan alamiah yang terjadi antara entitas. Pada umumnya penghubung (Relationship) diberi nama dengan kata kerja dasar, sehingga memudahkan untuk melakukan pembacaan relasinya (bisa dengan kalimat aktif atau kalimat pasif). Penggambaran hubungan yang terjadi adalah sebuah bentuk belah ketupat dihubungkan dengan dua bentuk empat persegi panjang.

4. Simbol State Diagram

Simbol	Deskripsi
	State Nilai atribut dan nilai link pada suatu waktu tertentu, yang memiliki suatu objek.
	Transisi Menggambarkan tujuan dari suatu state lainnya.
	Loop (Perulangan) Menggambarkan adanya perulangan pada suatu state.
	Start Point Merupakan permulaan dari sebuah sistem yang akan dikerjakan
	End Point Menggambarkan akhir dari sebuah sistem yang telah dikerjakan

5. Simbol Flowchart



DAFTAR ALGORITME

Algoritme 3.1 Pengumpulan Data <i>Scraping</i>	25
Algoritme 3.2 Algoritme <i>Preprocessing</i>	27
Algoritme 3.3 Seleksi <i>Input Layer</i>	29
Algoritme 3.4 <i>Backpropagation</i>	31
Algoritme 3.5 Prediksi <i>Feed Forward</i>	33
Algoritme 3.6 Menu <i>Monitoring</i> dan <i>Live Stream</i>	35
Algoritme 3.7 Menu Perbarui Bobot	36



DAFTAR ISI

ABSTRAK.....	iii
SURAT PERNYATAAN TIDAK PLAGIAT DAN PERSETUJUAN PUBLIKASI	ii
KATA PENGANTAR	iii
DAFTAR TABLE.....	v
DAFTAR GAMBAR	vi
DAFTAR SIMBOL	vii
DAFTAR ALGORITME.....	x
DAFTAR ISI.....	xi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	1
1.3 Batasan Masalah.....	1
1.4 Tujuan Penulisan.....	2
1.5 Metode Penelitian.....	2
1.6 Sistematika Penulisan.....	3
BAB 2 LANDASAN TEORI	4
2.1 Tanah Longsor	4
2.2 Media Sosial.....	4
2.2.1 Twitter.....	4
2.3 <i>Text Classification</i>	5
2.4 <i>Stemming</i>	5
2.5 <i>Feature Selection</i>	6
2.6 <i>Artificial Neural Network</i> (Jaringan Syaraf Tiruan)	6
2.6.1 Bag of Words	8
2.6.2 Pengaturan Weight (Bobot)	8
2.6.3 Activation Function (Fungsi Aktivasi)	9
2.6.4 Least Mean Square.....	9
2.6.5 Metode Backpropagation.....	10
2.7 Literature Review	13
BAB 3 ANALISIS MASALAH DAN PERANCANGAN SOLUSI	22
3.1 Analisa Masalah	22
3.2 Spesifikasi <i>Database</i>	22
3.3 Penyelesaian Masalah	23

3.4 Algoritme dan <i>Flowchart</i>	24
3.4.1 Pengumpulan Data Scraping.....	24
3.4.2 Preprocessing	26
3.4.3 Seleksi Input Layer	27
3.4.4 Training Data (Backpropagation)	29
3.4.5 Feed Forward (Prediksi)	32
3.4.6 Menu Monitoring dan Streaming Live Data.....	34
3.4.7 Menu Update Bobot Sinapsis	36
3.5 Simulasi Alur dan Perhitungan Algoritme	36
3.5.1 Menentukan Input Layer.....	36
3.5.2 Proses Training	38
3.6 Rancangan Layar.....	46
3.6.1 Rancangan Layar List Dataset	46
3.6.2 Rancangan Layar Prediksi	47
3.6.3 Rancangan Layar Halaman Prediksi Spesifik Data	48
BAB 4 IMPLEMENTASI DAN UJI COBA SOLUSI.....	49
4.1 Spesifikasi Sistem	49
4.1.1 Spesifikasi Perangkat Keras (Hardware)	49
4.1.2 Spesifikasi Perangkat Lunak (Software).....	49
4.2 Pengumpulan Data (<i>Web Scraping</i>)	49
4.3 Tahap Pelatihan Data (<i>Training Data</i>).....	53
4.4 Tampilan Layar	54
4.4.1 Halaman Utama – List Dataset	54
4.4.2 Halaman Prediksi Kelas	55
4.4.3 Halaman Prediksi Data yang Dipilih	57
4.5 Pengujian.....	57
4.5.1 Menghitung MSE (Mean Square Error)	57
4.5.2 Menghitung Akurasi Algoritme Backpropagation	59
BAB V PENUTUP	61
5.1 Kesimpulan	61
5.2 Saran.....	61
DAFTAR PUSTAKA	62
LAMPIRAN.....	64

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bencana adalah peristiwa atau rangkaian peristiwa yang mengancam dan mengganggu kehidupan dan penghidupan masyarakat yang disebabkan, baik oleh faktor alam dan/atau faktor nonalam. Salah satunya tanah longsor, yaitu salah satu jenis gerakan massa tanah atau batuan, ataupun percampuran keduanya, menuruni atau keluar lereng akibat terganggunya kestabilan tanah atau batuan penyusun lereng. Berdasarkan data dari BNPB, dalam 10 tahun terakhir, telah terjadi sebanyak 4.971 kasus tanah longsor di Indonesia, dengan total korban terdampak dan menungsi sebanyak 216.028 orang.

Berkembangnya teknologi dan informasi, memudahkan masyarakat untuk saling berinteraksi, dan bertukar informasi melalui media sosial, seperti Twitter. Hal ini sangat berperan penting dalam persebaran informasi secara luas dan cepat, mengingat besarnya jumlah pengguna Twitter aktif di Indonesia, akan sangat membantu jika masarakat dapat turut aktif memberikan dukungan berupa informasi kejadian bencana melalui media sosial Twitter. Dengan begitu pihak pemerintah dapat dengan cepat mengambil tindakan dan pertolongan kepada korban di lokasi bencana.

Twitter menjadi salah satu media sosial yang paling banyak digunakan oleh masyarakat. Dengan adanya media sosial Twitter, informasi dapat dengan cepat tersebar, termasuk di dalamnya *tweet – tweet* mengenai informasi bencana tanah longsor. Tercatat ada sekitar 5600 *tweet* dengan kata kunci “longsor” yang didapat dari tahun 2016 s/d 2019.

Penulis mencoba membuat implementasi algoritme *Artificial Neural Network (ANN)*, yang akan mengklasifikasi secara otomatis data-data *tweet* dengan kata kunci “longsor” yang didapat dari Twitter, guna mengindikasi *tweet* yang memang mengandung peringatan atau laporan kejadian tanah longsor yang sedang terjadi. Pengklasifikasian tersebut berdasarkan bobot tiap kata yang terdapat dalam setiap *tweet*. Penulis akan menerapkan implementasi algoritme pengklasifikasian ini pada BNPB.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang diatas, maka dapat dirumuskan masalah sebagai berikut :

1. Bagaimana cara mengklasifikasi *tweet* untuk mendapatkan informasi peringatan kejadian tanah longsor pada implementasi algoritme *Backpropagation Artificial Neural Network (ANN)* ?
2. Apakah klasifikasi teks yang dihasilkan akurat ?

1.3 Batasan Masalah

Adapun batasan masalah dari penulisan tugas akhir ini sebagai berikut :

- a. Aplikasi ini menggunakan bahasa pemrograman Python.
- b. *Platform* yang digunakan hanya berbasis *web*.
- c. Dataset yang digunakan, terbatas pada *tweet* dengan kata kunci “longsor”, dalam kurun waktu 2015 s/d 2019.
- d. Dataset yang dilatih hanyalah *tweet* berbahasa Indonesia yang mengandung kata kunci “longsor”.
- e. Aplikasi yang dibuat hanya mngklasifikasikan *tweet* menjadi dua kategori, yaitu : “warning” dan “non-warning”.

1.4 Tujuan Penulisan

Tujuan dari penulisan adalah untuk mengklasifikasi *tweet* yang kirimkan oleh pengguna Twitter di Indonesia, guna mendapatkan informasi peringatan atau laporan kejadian bencana tanah longsor melalui deteksi media sosial Twitter, serta menguji keakuratan pengklasifikasian teks dengan metode *Backpropagation Artificial Neural Network*.

1.5 Metode Penelitian

Dalam tugas akhir ini ada beberapa metode yang digunakan untuk memperoleh informasi dan menyelesaikan masalah yang ditemui. Adapun metode – metode ini sebagai berikut:

- a. Studi Literatur
Kegiatan ini dilakukan dengan mengumpulkan, membaca dan mempelajari dokumen-dokumen, jurnal ilmiah, artikel internet dan lain-lain. Tujuan studi literatur adalah untuk memperoleh informasi yang akan dijadikan refensi dalam penulisan tugas akhir.
- b. Metode Pengembangan
 - 1) Analisis
Pada tahap ini penulis akan melakukan analisis bagaimana menerapkan algoritme *Artificial Neural Network* untuk mengklasifikasi data-data yang didapat dari Twitter ke dalam kelas-kelas yang telah ditentukan, berdasarkan bobot dari masing-masing kata yang terkandung di setiap *tweet*.
 - 2) Desain
Pada tahap ini merancang alur program yang akan dibuat serta membuat rancangan design awal.
 - 3) Implementasi
Pada tahap ini akan dilakukan pengkodean yang menggunakan bahasa pemrograman Python sebagai inti dari proses algoritme klasifikasi *Artificial Neural Network*, penyajian program berbasiskan web menggunakan HTML, CSS, dan Javascript. PHPStorm digunakan sebagai *text editor* untuk membuat program serta MySQL digunakan sebagai database.
 - 4) Pengujian

Pada tahap ini dilakukan pengujian terhadap aplikasi yang telah dibuat, apakah berjalan sesuai dengan tujuan penelitian atau tidak.

1.6 Sistematika Penulisan

BAB I : PENDAHULUAN

Dalam bab ini membahas mengenai pendahuluan, latar belakang penulisan, masalah yang dijelaskan dengan uraian singkat, tujuan penulisan yang menggambarkan tentang apa saja yang ingin dicapai, batasan masalah yang menjelaskan tentang lingkup permasalahan yang dibahas dan penyelesaian yang dilakukan, metoda pengembangan dalam pengumpulan data, dan sistematika penulisan yang menjelaskan uraian singkat per babnya.

BAB II : LANDASAN TEORI

Pada bab ini menjelaskan teknik pembuatan sistem yang diambil dari berbagai sumber seperti jurnal, buku-buku, artikel internet. Serta terdapat juga algoritme-algoritme yang dipakai.

BAB III : ANALISA MASALAH DAN RANCANGAN PROGRAM

Dalam bab ini, menjelaskan tentang pendahuluan, permasalahan dan strategi pemecahan masalah, program aplikasi, analisa dan perancangan program berupa aktifitas dan *activity diagram*, rancangan layar, data dan basis data, operasi dari prosedur yang berisi *flowchart*, dan algoritme.

BAB IV : IMPLEMENTASI DAN UJICOBA

Pada bab ini menjelaskan implementasi dari algoritme yang dipakai dalam sistem dan uji coba program. Di bab ini juga membahas tentang spesifikasi untuk menggunakan sistem ini, serta tampilan program yang sebenarnya.

BAB V : PENUTUP

Dalam bab ini, berisi tentang kesimpulan yang didapat dari hasil analisa bab-bab sebelumnya, dan saran yang dapat berguna bagi pengembangan aplikasi ini selanjutnya

BAB 2

LANDASAN TEORI

2.1 Tanah Longsor

Menurut Undang-undang Nomor 24 Tahun 2007 Tentang Penanggulangan Bencana seperti yang terdapat di laman resmi BNPB (Badan Nasional Penanggulangan Bencana), bencana alam dapat dapat didefinisikan sebagai peristiwa atau rangkaian peristiwa yang mengancam dan mengganggu kehidupan dan penghidupan masyarakat yang disebabkan, baik oleh faktor alam dan/atau faktor nonalam maupun faktor manusia sehingga mengakibatkan timbulnya korban jiwa manusia, kerusakan lingkungan, kerugian harta benda, dan dampak psikologis.

Diantara macam-macam bencana alam yang ada, salah satunya adalah tanah longsor, yaitu salah satu jenis gerakan massa tanah atau batuan, ataupun percampuran keduanya, menuruni atau keluar lereng akibat terganggunya kestabilan tanah atau batuan penyusun lereng (BNPB, Definisi dan Jenis Bencana, 2017).

Tanah longsor terjadi sebanyak 246 kali dalam kurun waktu sepuluh tahun terakhir. Tanah longsor juga menyebabkan kerusakan rumah paling parah dibandingkan bencana-bencana alam lainnya, yaitu sebanyak 800 rumah rusak berat, 258 rumah rusak sedang, dan 997 rumah rusak ringan dalam kurun waktu yang sama (BNPB, Data Informasi Bencana Indonesia (DIBI), 2017).

2.2 Media Sosial

Media sosial adalah media berbasis internet yang mana memungkinkan para pengguna untuk dapat berpartisipasi, dan berbagi konten nya masing-masing, termasuk di dalamnya yaitu blog, wiki, forum, dsb. (Putri dkk., 2016)

Media sosial memiliki dampak yang besar terhadap perubahan perilaku dan aktivitas masyarakat, salah satu dampak positif dari media sosial yaitu penyebaran informasi dapat berlangsung dengan cepat. Siapapun dapat menyebarluaskan informasi baru kapan saja, dan juga siapapun dapat mendapatkan informasi yang tersebar di media sosial kapan saja, tanpa terbatasi waktu dan ruang (Anang, 2016)

2.2.1 Twitter

Twitter adalah jejaring sosial dan layanan *microblogging*, memungkinkan pengguna terdaftar untuk membaca dan memposting pesan singkat, yang disebut *tweet*. Pesan Twitter dibatasi hingga 280 karakter dan pengguna juga dapat mengunggah foto atau video pendek. *Tweet* dapat dikirim secara umum atau dikirim sebagai pesan langsung ke pengguna lain. Twitter adalah salah satu jejaring sosial paling populer di seluruh dunia. Tercatat

sebanyak 330 juta pengguna aktif Twitter dari berbagai negara pada kuartal pertama 2019 ini. (Statista Research Department, 2019)

Twitter menjadi salah satu media sosial yang paling populer dan paling sering digunakan. Sehingga informasi apapun hampir dapat kita temukan semua di Twitter. Senada dengan itu, Twitter sendiri mempunyai tagline “*Twitter is what's happening in the world and what people are talking about right now.*” dan juga “*When it happens, it happens on Twitter.*” (Twitter, 2019).

Twitter menjadi sumber pengumpulan data yang baik, karena masifnya jumlah dataset yang dapat diperoleh, menjadikan Twitter sebagai sumber dataset yang digunakan.

2.3 *Text Classification*

Klasifikasi adalah bentuk *supervised machine learning* yang bertujuan untuk mengidentifikasi kategori yang dimiliki oleh sekumpulan teks yang diberikan, berdasarkan *training dataset* yang telah ditentukan sebelumnya (Suharno dkk., 2017). Mengklasifikasikan dokumen ke dalam sebuah domain tertentu menjadi sangat penting karena sebagian besar *problem set* yang dihadapi dalam kasus-kasus *text mining* lainnya, seperti *text summarization*, *information extraction*, mencari hubungan semantik, dll., hanya bisa mencapai hasil maksimal jika diaplikasikan dalam domain tertentu yang telah ditentukan.

Menurut (Zhang *et al.*, 2008), tahap-tahap pada klasifikasi teks yaitu diawali dengan pengumpulan data (teks), dengan proporsi pembagian jumlah data yang digunakan sesuai keperluan sebesar 2/3 digunakan sebagai *training data* dan 1/3 digunakan sebagai *test data* dengan pengambilan sampel data acak. Selanjutnya, data-data melalui tahap *preprocessing* meliputi penghapusan kata-kata *stop words*, dan proses *stemming*. Penghapusan kata-kata *stop words* berfungsi untuk memilah kata-kata yang dianggap fungsional, dan mengeliminasi kata-kata yang tidak memiliki arti. Menghapus kata-kata *stop words* juga dapat memperkecil dimensi *input layer* karena cenderung sering terulang dalam teks, serta dapat membuat proses *training* menjadi lebih efisien. Teks kemudian melalui tahap *stemming*, yaitu tahap untuk mengembalikan kata menjadi bentuk dasarnya.

2.4 *Stemming*

Stemming adalah proses pengembalian sebuah kata menjadi kata dasarnya. *Stemming* bekerja dengan mengilangkan imbuhan dan akhiran kata, penghilangan imbuhan dan akhiran kata sangat berperan penting dalam *machine translation*, *document summarization* dan *text classification* (Nazief *et al.*, 2005) .

Word stemming digunakan pada tahap *preprocessing*, untuk mendapatkan teks yang efisien dan efektif dari *dataset* yang tersedia.

2.5 Feature Selection

Pemilihan fitur, bertujuan untuk mengurangi dimensi dari data yang akan diolah sehingga meningkatkan efektifitas dari implementasi algoritme klasifikasi. Pada penelitian ini, pemilihan fitur dilakukan pada tahap *preprocessing*.

Pemilihan fitur digunakan pada klasifikasi teks yang memiliki dimensi yang sangat besar. Teknik ini berkerja dengan memilih hanya fitur yang telah ditentukan yang akan digunakan untuk melatih dan menguji pengklasifikasi.(Zareapoor *et al.*, 2015)

Penelitian ini menggunakan *stoplist* yang dibuat oleh (Tala, 2017) sebagai acuan pemilihan fitur, berdasarkan penelitiannya yang telah merumuskan kumpulan kata yang termasuk ke dalam *stoplist* berdasarkan jumlah frekuensi kemunculan kata tersebut.

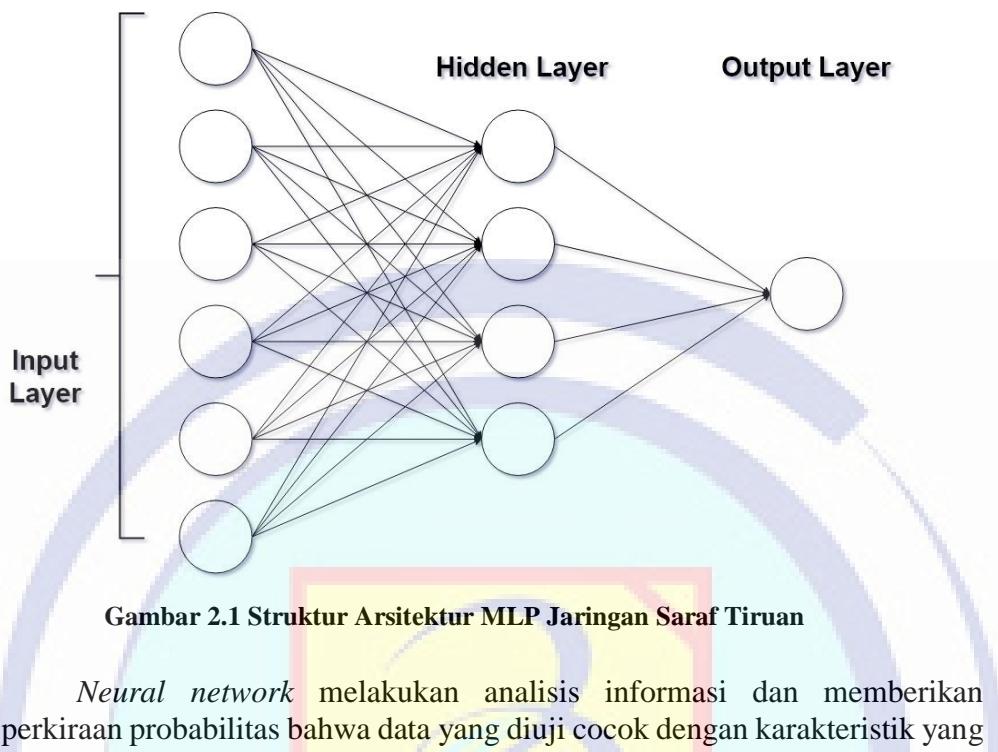
2.6 Artificial Neural Network (Jaringan Syaraf Tiruan)

Jaringan Syaraf Tiruan (JST) merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologi. Cara kerja dari JST ini seperti dengan sistem kerja jaringan makhluk hidup yang memiliki neuron-neuron jaringan syaraf (Ahmad dkk., 2019). Senada dengan apa yang disampaikan oleh Ahmad dkk., menurut Riswanto, Sutikno dan Indriyati (2014), Jaringan Saraf Tiruan (JST) merupakan suatu bentuk arsitektur yang terdistribusi paralel dengan sejumlah besar titik dan hubungan antar titik tersebut. Tiap titik berhubungan dari satu titik ke titik lain mempunyai harga yang diasosiasikan sebagai bobot. Setiap titik memiliki nilai yang diasosiasikan sebagai nilai aktivasi titik. Kemampuan yang dimiliki JST dapat digunakan untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh atau input yang dimasukkan dan membuat prediksi tentang kemungkinan output yang akan muncul .

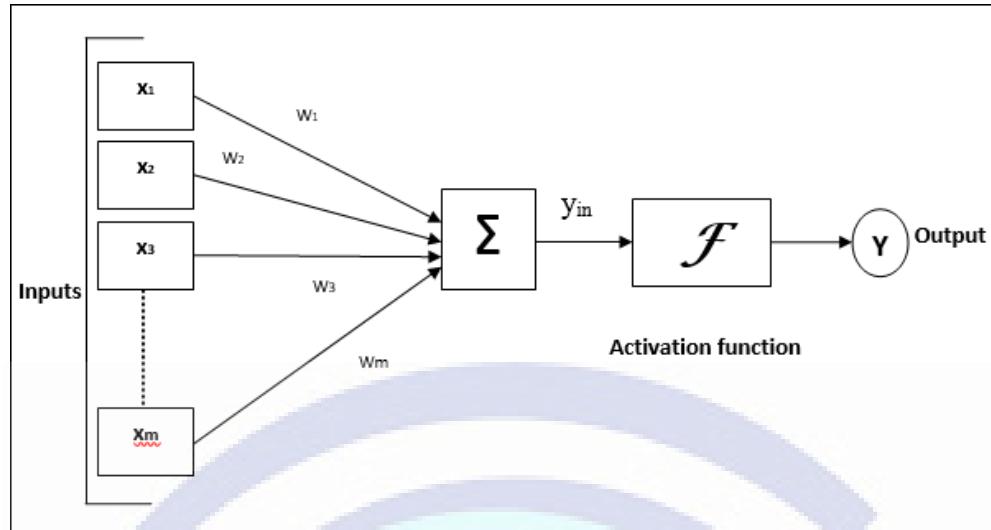
Struktur jaringan saraf tiruan memiliki beberapa elemen pemrosesan tunggal, yang disebut neuron. Neuron terhubung satu sama lain oleh sebuah *joint* (penghubung) yang telah diberikan nilai berupa *weight* (bobot). Jaringan saraf tiruan terdiri dari beberapa lapis (*layer*), atau biasa disebut dengan MLP (*Multi Layer Perceptron*). MLP adalah pendekatan umum untuk menyelesaikan *problem set* berupa regresi. MLP memiliki tiga lapisan: lapisan *input*, lapisan *output*, dan lapisan tersembunyi (*hidden layer*) (Moghaddam *et al.*, 2016).

Lapisan input pada *Multi Layer Perceptron*, hanya menerima informasi dan bertindak seperti variabel independen. Dengan demikian, jumlah neuron pada lapisan input ditentukan berdasarkan masalah yang dihadapi (*problem set*) dan tergantung pada jumlah variabel independen. Lapisan output bertindak sebagai variabel dependen dan jumlah neuronnya tergantung pada jumlah variabel dependen. Namun, tidak seperti lapisan *input* dan lapisan *output*, layer tersembunyi (*hidden layer*) tidak mewakili konsep apa pun dan hanya merupakan hasil antara dalam proses penghitungan nilai *output* (Askari *et al.*, 2017).

Secara garis besar, jaringan saraf tiruan dapat dilihat pada Gambar 2.1.



Neural network melakukan analisis informasi dan memberikan perkiraan probabilitas bahwa data yang diuji cocok dengan karakteristik yang telah terlebih dahulu dilatih dan dijadikan acuan. Sedangkan probabilitas kecocokan yang dihasilkan oleh *neural network* dapat mencapai 100%, keakuratan klasifikasi sepenuhnya bergantung pada hasil dari *training data* yang didapat sistem berdasarkan analisis dari data uji yang telah ditentukan. *Neural network* mendapatkan pengalaman pada awalnya dengan melatih sistem untuk mengidentifikasi dengan benar contoh-contoh masalah yang dipilih sebelumnya. Hasil pelatihan data pada *neural network* ditinjau, lalu konfigurasi sistem disempurnakan hingga analisis *neural network* terhadap data pelatihan mencapai tingkat yang memuaskan. Selain pada pelatihan awal, jaringan saraf juga mendapatkan pengalaman dari waktu ke waktu karena melakukan analisis pada data yang terkait dengan masalah tersebut. Diagram alur proses *artificial neural network* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Diagram proses Neural Network

2.6.1 *Bag of Words*

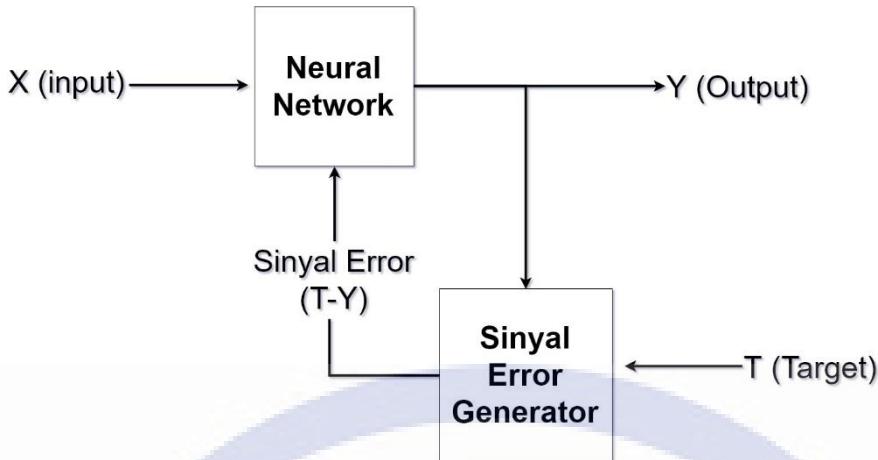
Kata-kata adalah inti dari klasifikasi teks. Oleh karenanya, model *bag-of-words*, dapat memberikan *baseline* yang lebih akurat dan efektif dibandingkan dengan model yang meliputi representasi fitur yang lebih kompleks. (Jin *et al.*, 2016)

Cara kerja *bag-of-words* adalah dengan memanfaatkan serangkaian kata-kata menonjol yang mewakili masing-masing kelas dokumen. Misalnya, kata "pelatih", "sepakbola", "bola basket" muncul relatif sering dalam olahraga, dan kata-kata "chipset", "compiler" dan "Linux" relatif sering muncul untuk kategori teknologi informasi.

Konsep ini digunakan sebagai *input layer*, sehingga membuat jumlah neuron yang digunakan pada *input layer* lebih efektif dan tidak terlalu besar. Sehingga dapat mempercepat proses *training data*, dan meningkatkan akurasi dari prediksi *class* dari data yang diuji.

2.6.2 *Pengaturan Weight (Bobot)*

Makna "belajar", dalam jaringan saraf tiruan, adalah metode memodifikasi bobot koneksi antara neuron jaringan tertentu. Pada penelitian kali ini, penulis menggunakan model *supervised learning*, sehingga proses pengaturan bobot dapat dilihat seperti pada Gambar 2.3.

Gambar 2.3 Diagram pengaturan bobot pada *Supervised Learning*

Proses *backpropagation* dimulai dengan memberikan bobot pada tiap layer dengan sebuah nilai acak, baru selanjutnya bobot akan mengalami modifikasi dan penyesuaian seiring berjalannya proses *training data*. *Backpropagation* akan memperbarui nilai bobot, sesuai dengan perhitungan karakteristik data latih yang telah diberikan label sebelumnya, hingga mencapai titik keakuratan maksimum.

2.6.3 Activation Function (*Fungsi Aktivasi*)

Fungsi aktivasi dapat didefinisikan sebagai upaya yang diterapkan pada input untuk mendapatkan output yang tepat. Pada penelitian ini, penulis menggunakan *sigmoid activation function*.

Sigmoid activation function ini melakukan optimasi input dengan mengubahnya menjadi bernilai antara 0 dan 1. Nilainya selalu positif dan selalu dibatasi, yang berarti outputnya tidak boleh kurang dari 0 dan lebih dari 1 (Kartika, Irawan dan Triyanto, 2016). Rumus fungsi aktivasi sigmoid adalah sebagai berikut :

$$f(x) = \frac{1}{1 + e^{-x}} \dots \dots \dots (1)$$

Dengan rumus turunan dari fungsi tersebut yaitu :

$$f'(x) = x * (1 - x) \dots \dots \dots (2)$$

2.6.4 Least Mean Square

Diperkenalkan pertama kali oleh (Haykin *et al.*, 2003) bertujuan untuk meminimalisir nilai *error* di setiap pola pelatihan data. Cara kerjanya menyerupai algoritme *supervised learning*

dengan mengaplikasikan fungsi aktivasi yang terus menerus dan kontinu.

Konsep dasar *least mean square* adalah dengan pendekatan *gradient-descent*, yang mana secara kontinu dilakukan terus menerus, yaitu dengan memperbarui bobot sinaptik setiap terjadi selisih antara hasil yang didapat dengan hasil yang diekspektasikan. Rumus *least means square* adalah sebagai berikut :

$$\Delta\omega_i = \alpha \cdot x_i \cdot e_j \dots \dots \dots \quad (3)$$

Keterangan :

$\Delta\omega_i$ = perubahan bobot yang ke- i ;

α = *learning rate* yang bersifat positif dan konstan;

x_i = nilai masukan dari neuron pada *layer* sebelumnya;

$e_j = (t - y_{in})$, selisih dari target (hasil yang diinginkan) dengan hasil sebenarnya (y_{in});

Proses pembaruan nilai bobot dapat dilakukan dengan mengikuti dua aturan berikut, jika $t \neq y$ (target tidak sama dengan hasil *output*) maka nilai bobot yang baru adalah hasil dari bobot lama dijumlah dengan perubahan bobot, atau

$$\omega_{baru} = \omega_{lama} + \Delta\omega \dots \dots \dots \quad (4)$$

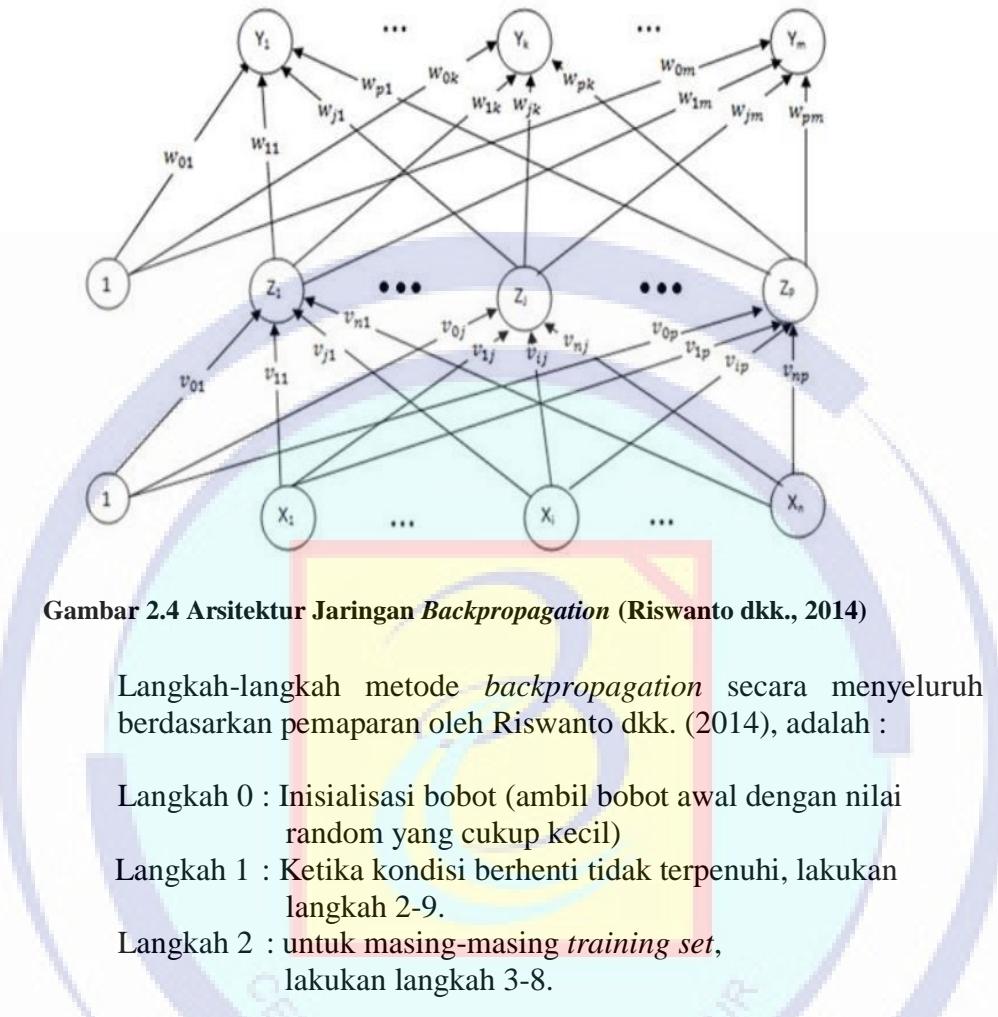
Sementara jika $t = y$ (target sama dengan hasil *output*) maka nilai bobot tidak mengalami perubahan.

2.6.5 Metode Backpropagation

Backpropagation merupakan metode yang termasuk ke dalam algoritme *supervised learning*. *Backpropagation* bekerja dengan memberikan sepasang pola pada jaringan, yang mana pola tersebut terdiri atas pola masukan dan pola keluaran yang diinginkan. *Backpropagation* kemudian akan menyesuaikan bobot dari tiap pola masukan, agar menyamai sedekat mungkin dengan hasil yang diinginkan, melalui perhitungan selisih hipotesis dengan keluaran yang diinginkan tersebut. Rumusan bobot yang telah didapatkan melalui proses propagasi balik tersebut, dapat digunakan pada masukan baru, dan dengan itu dapat dilakukan prediksi (peramalan) berdasarkan hasil pelatihan sebelumnya. Semakin kecil selisih dari hasil *backpropagation* dengan hasil yang diinginkan, artinya semakin akurat prediksi yang dapat dihasilkan (Riswanto dkk., 2014).

Tahapan dari *backpropagation* terbagi menjadi : *feed-forward* pola masukan, *backpropagation* terhadap *error* (selisih

hasil hipotesis dan keluaran target), dan penyesuaian bobot. Arsitektur jaringan *backpropagation* dapat dilihat pada Gambar 2.4.



Feedforward:

Langkah 3 : Tiap-tiap *input neuron* (X_i , $i=1,2,3,\dots,n$) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada *layer* selanjutnya (*hidden layer*)

Langkah 4 : Tiap-tiap *hidden neuron* ($Z_i, j=1,2,3,\dots,p$) menjumlahkan sinyal-sinyal dari input yang telah terbobot :

$$z_in_j = V_{Oj} + \sum_{i=1}^n X_i V_{ij} \dots \dots \dots \quad (5)$$

Kemudian gunakan fungsi aktivasi (1) untuk menghitung sinyal *output*:

$$Z_j = f(z_{in_j}) \dots \dots \dots \quad (6)$$

Hitung fungsi aktivasi untuk menghitung sinyal keluaran, $Z_j = f(z_{in_j})$, kemudian sinyal ini dikirim kesemua *neuron* pada *layer* selanjutnya (*output layer*).

Langkah 5 : Tiap-tiap *output neuron* (Y_k , $K=1,2,3,\dots,m$) menjumlahkan sinyal-sinyal input terbobot.

$$Y_in_k = W_{Ok} + \sum_{i=1}^p Z_i W_{jk} \dots \dots \dots \quad (7)$$

gunakan fungsi aktivasi seperti yang tertera pada rumus (2) untuk menghitung sinyal *output* :

$$Y_k = f(Y_{in_k}) \dots \dots \dots \quad (8)$$

Kemudian sinyal tersebut dikirim ke semua *neuron* pada lapisan selanjutnya (*output neuron*).

Backpropagation:

Langkah 6 : Tiap-tiap *output neuron* (Y_k , $K=1, 2, 3, \dots, m$) menerima target pola yang berhubungan dengan pola input pembelajaran, hitung informasi *error*-nya, dengan mengalikan selisih antara nilai target dan nilai prediksi dengan hasil dari fungsi turunan aktivasi sigmoid (2) :

$$\delta_k = (t_k - Y_k) * f'(Y_{in_k}) \dots \dots \dots \quad (9)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai W_{jk}):

$$\Delta W_{ik} = \alpha \delta_k z_i \dots \dots \dots \quad (10)$$

hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai W_{ok}):

$$\Delta W_{\theta k} = \alpha \delta_k \dots \dots \dots \dots \quad (11)$$

kemudian, nilai δ_k diteruskan ke *neuron* pada *layer* sebelumnya (mundur).

Langkah 7 : Tiap-tiap unit tersembunyi (Z_j , $j=1,2,3,\dots,p$) menjumlahkan delta *input*-nya (dari unit-unit yang berada pada lapisan setelahnya):

$$\delta_{in_j} = \sum_{k=1}^m \delta_k W_{jk} \dots \dots \dots \quad (12)$$

Kalikan nilai ini dengan turunan dari fungsi aktivasinya (2) untuk menghitung informasi error:

$$\delta_j = \delta_{in_j} * f'(z_{in_j}) \dots \dots \dots \quad (13)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai V_{ij}):

$$\Delta V_{jk} = \alpha \delta_j X_i \dots \dots \dots \quad (14)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai V_{0j}):

$$\Delta V_{0j} = \alpha \delta_j \dots \dots \dots \quad (15)$$

Langkah 8 : Tiap-tiap *output neuron* (Y_k , $k=1,2,3,\dots,m$) akan memperbaiki bias dan bobotnya ($i=1,2,3,\dots,p$):

$$W_{jk}(\text{baru}) = W_{jk}(\text{lama}) + \Delta W_{ij} \dots \dots \dots \quad (16)$$

Tiap-tiap unit tersembunyi (Z_j , $j=1,2,3,\dots,p$) memperbaiki bias dan bobotnya ($i=0,1,2,\dots,n$) :

$$V_{jk}(\text{baru}) = V_{jk}(\text{lama}) + \Delta V_{ij} \dots \dots \dots \quad (17)$$

Langkah 9 : Uji syarat berhenti.

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola dengan menggunakan fase propagasi maju saja (langkah 3 dan 4) untuk menentukan keluaran jaringan.

2.7 Literature Review

Penulis	Wen Zhang, Taketoshi Yoshida, dan Xijin Tang
No. ISSN	09507051
Terbitan	<i>Knowledge-Based Systems</i>

Tahun	2008
Judul	<i>Text Classification Based on Multi-word with Support Vector Machine</i>
Deskripsi	<p>Banyaknya upaya yang telah diinvestasikan pada topik penambangan data untuk memperkaya representasi teks menggunakan model <i>Vector Space Model</i> (VSM) untuk meningkatkan kinerja teknik penambangan teks seperti klasifikasi teks dan pengelompokan teks. Perhatian utama dalam makalah ini adalah untuk menyelidiki efektivitas penggunaan multi-kata untuk representasi teks pada kinerja klasifikasi teks. Pertama, metode praktis diusulkan untuk mengimplementasikan ekstraksi multi-kata dari dokumen berdasarkan struktur sintaksis. Kedua, dua strategi sebagai representasi konsep umum dan representasi subtopik disajikan untuk mewakili dokumen menggunakan multi-kata yang diekstraksi. Secara khusus, <i>k-mismatch</i> dinamis diusulkan untuk menentukan keberadaan multi-kata panjang yang merupakan subtopik dari isi dokumen. Akhirnya, kami melakukan serangkaian percobaan pada klasifikasi dokumen Reuters-21578 menggunakan representasi dengan multi-kata. Kami menggunakan kinerja representasi dalam kata-kata individual sebagai baseline, yang memiliki dimensi terbesar dari fitur yang ditetapkan untuk representasi tanpa preprocessing linguistik. Selain itu, kernel linear dan kernel polinomial non-linear dalam <i>Support Vector Machine</i> (SVM) diperiksa komparatif untuk klasifikasi untuk menyelidiki efek dari jenis kernel pada kinerja mereka. Istilah indeks dengan <i>information gain</i> (IG) rendah dikeluarkan dari fitur yang ditetapkan pada persentase yang berbeda untuk mengamati kekokohan masing-masing metode klasifikasi. Eksperimen kami menunjukkan bahwa dalam representasi multi-kata, representasi subtopik mengungguli representasi konsep umum dan kernel linier mengungguli kernel non-linear SVM dalam mengklasifikasikan data Reuters. Pengaruh penerapan strategi representasi yang berbeda lebih besar daripada efek penerapan kernel SVM yang berbeda pada kinerja klasifikasi. Lebih jauh lagi, representasi yang menggunakan kata-kata individual mengungguli representasi apa pun menggunakan multi-kata. Ini konsisten dengan pendapat utama mengenai peran preprocessing linguistik pada fitur dokumen ketika menggunakan SVM untuk klasifikasi teks.</p>
Penulis	Simon Haykin, dan Bernard Widrow

No. ISBN	978-0471215707
Terbitan	Wiley Online Library
Tahun	2003
Judul	Least Mean Square Adaptive Filters
Deskripsi	<p>Di bidang teknik listrik dan pemrosesan sinyal, beberapa algoritma telah terbukti dapat beradaptasi sebagai algoritma least-mean-square (LMS). Diciptakan oleh Bernard Widrow dan M. Hoff, algoritma sederhana namun efektif ini sekarang merupakan landasan untuk desain filter transversal adaptif (<i>tapped-delay-line</i>). Saat ini, bekerja secara efisien dengan filter adaptif LMS tidak hanya melibatkan memahami dasar-dasar mereka, itu juga berarti tetap terkini dengan banyak aplikasi mereka dalam sistem praktis. Namun, tidak ada sumber daya tunggal yang menyajikan pemeriksaan terkini dari semua ini dan semua aspek penting lainnya dari filter LMS - sampai sekarang. <i>Least-Mean-Square Adaptive Filter</i> menawarkan tampilan paling definitif pada filter LMS yang tersedia di mana saja. Di sini, pembaca akan mendapatkan perspektif yang meyakinkan tentang properti yang diinginkan yang membuat LMS memfilter teknologi <i>turnkey</i> untuk pemrosesan sinyal adaptif. Sama pentingnya, <i>Filter Adaptif Least-Mean-Square</i> menyatakan kontribusi para ahli terkenal yang wawasannya mencerminkan <i>state-of-the-art</i> saat ini.</p>

Penulis	Kaveh Ostad Ali Askari, Mohammad Shayannejad, dan Hossein Ghorbanizadeh Kharazi
No. ISSN	19763808
Terbitan	KSCE Journal of Civil Engineering
Tahun	2017
Judul	<p><i>Artificial Neural Network for Modeling Nitrate Pollution of Groundwater in Marginal Area of Zayandeh-rood River, Isfahan, Iran</i></p> <p>(Jaringan Syaraf Tiruan untuk Pemodelan Polusi Nitrat Air Tanah di Area Marjinal Sungai Zayandeh-rood, Isfahan, Iran)</p>
Deskripsi	Penggunaan berlebihan pupuk kimia, terutama pupuk nitrogen untuk meningkatkan panen dan pemurnian yang tidak tepat, dan pengiriman air limbah kota dan industri

	diusulkan sebagai faktor yang meningkatkan jumlah nitrat dalam air tanah di daerah ini. Dengan demikian, investigasi kontaminasi nitrat sebagai salah satu masalah lingkungan terpenting dalam air tanah diperlukan. Dalam penelitian ini, pemodelan dan estimasi polusi nitrat dalam air tanah area marginal Sungai Zayandeh-rood, Isfahan, Iran, diselidiki menggunakan kualitas air dan <i>artificial neural network</i> . 100 sumur (77 sumur pertanian, 13 sumur minum dan 10 kebun sumur) di daerah pinggiran Sungai Zayandeh-rood, Isfahan, Iran dipilih. Perangkat lunak MATLAB dan jaringan <i>Perceptron</i> tiga lapis digunakan. Aturan pembelajaran <i>backpropagation</i> dan fungsi aktivasi sigmoid diterapkan untuk proses pelatihan. Setelah sering berasperiments, jaringan dengan satu lapisan tersembunyi dan 19 neuron membuat kesalahan paling kecil dalam proses pelatihan jaringan, pengujian dan validasi. Model JST dapat diterapkan untuk penyelidikan parameter kualitas air.
Penulis	Abdullah Ahmad, Pipit Mutiara Putri, Winanda Alifah, Indra Gunawan, dan Solikhun
No. ISSN	2598-7542
Terbitan	Jurnal RESISTOR (Rekayasa Sistem Komputer)
Tahun	2019
Judul	Analisis Jaringan Syaraf Tiruan Metode Backpropogation Dalam Memprediksi Ketersediaan Komoditas Beras Berdasarkan Provinsi Di Indonesia
Deskripsi	Pangan merupakan kebutuhan utama manusia yang harus dipenuhi setiap saat. Hak tersebut merupakan salah satu hak asasi manusia, sebagaimana dalam pasal 27 UUD 1945 maupun dalam Deklarasi Roma (1996). Pertimbangan tersebut mendasari terbitnya UU No. 7/1996 tentang Pangan. Dengan pertimbangan tersebut, Pemerintah selalu berupaya untuk meningkatkan ketahanan pangan yang bersumber dari peningkatan produksi dalam negeri. Penelitian ini diharapkan bisa memberikan kontribusi bagi pemerintah agar dapat memprediksi ketersediaan komoditas beras berdasarkan provinsi di Indonesia ke depannya. Data yang digunakan adalah data dari Badan Statistik Nasional melalui website www.bps.go.id . Data tersebut adalah data produksi padi/beras berdasarkan provinsi di Indonesia dalam rentang waktu tahun 2010 sampai dengan tahun 2015. Algoritme yang digunakan pada penelitian ini adalah Jaringan Saraf Tiruan dengan metode Backpropagation. Sebanyak 4 arsitektur yakni 4-4-1,

	4-8-1, 4- 16-1, 4-32-1. Keluaran yang dihasilkan adalah pola terbaik dari arsitektur JST. Model arsitektur terbaik adalah 4-4-1 dengan epoch 218, MSE 0.012728078 dan tingkat akurasi 97%. Dari model ini diperoleh prediksi ketersediaan komoditas beras berdasarkan provinsi di Indonesia.
--	--

Penulis	Ayu Kartika, Beni Irawan, dan Dedi Triyanto
No. ISSN	2338 – 493X
Terbitan	Jurnal Coding, Sistem Komputer Untan
Tahun	2016
Judul	Prediksi Wilayah Rawan Kebakaran Hutan dengan Metode Jaringan Syaraf Tiruan Propagasi Balik (Studi Kasus : Daerah Kabupaten Kuburaya)
Deskripsi	Jaringan Syaraf Tiruan merupakan salah satu sistem yang digunakan dalam memprediksi suatu kejadian. Dalam kasus ini Jaringan Syaraf Tiruan digunakan dalam memprediksi suatu kejadian wilayah rawan kebakaran hutan, dengan mengambil studi kasus daerah Kabupaten Kuburaya. Kasus ini, Jaringan Syaraf Tiruan menggunakan metode pembelajaran Backpropagation yang juga dikenal dengan propagasi balik. Jaringan Syaraf Tiruan digunakan data sebanyak satu bulan sebagai masukan yang merupakan faktor terjadinya kebakaran hutan, terdiri dari Luas Lahan Gambut, Radiasi Matahari, Kelembaban Nisbi, Suhu, Curah Hujan, Kecepatan Angin dan Titik Api digunakan sebagai target untuk kejadian pada bulan berikutnya . Prediksi dapat dilakukan melalui interface dan di visualisasikan melalui tampilan peta tematik. Dari hasil pengujian diperoleh akurasi dengan rata-rata persentase sebesar 71,50%. Dari hasil prediksi pertama diperoleh pula kasus terendah dengan jumlah 0 titik api yakni pada kecamatan Sungai Kakap, Rasau Jaya, Teluk Pakedai. Sedangkan hasil prediksi tertinggi terdapat pada kecamatan Sungai Raya dengan nilai prediksi sebesar 2 titik api.

Penulis	Peng Jin, Zhang Yue, Xingyuan Chen, dan Yunqing Xia
No. ISSN	10450823

Terbitan	IJCAI International Joint Conference on Artificial Intelligence
Tahun	2016
Judul	<i>Bag-of-Embeddings for Text Classification</i>
Deskripsi	Kata-kata adalah pusat klasifikasi teks. Telah ditunjukkan bahwa model-model Naive Bayes sederhana dengan fitur-fitur <i>word</i> dan <i>bigram</i> dapat memberikan akurasi yang sangat kompetitif bila dibandingkan dengan model-model yang lebih canggih dengan fitur-fitur <i>part-of-speech</i> , sintaksis dan <i>semantic</i> . <i>Embeddings</i> menawarkan fitur distribusi tentang kata-kata. Kami mempelajari model klasifikasi sederhana secara konseptual dengan memanfaatkan <i>multi-prototype embeddings</i> kata berdasarkan kelas teks. Asumsi utama adalah bahwa kata-kata menunjukkan karakteristik distribusi yang berbeda di bawah kelas teks yang berbeda. Berdasarkan asumsi ini, kami melatih representasi kata multi-prototipe distribusi untuk kelas teks yang berbeda. Diberikan dokumen baru, kelas teksnya diprediksi dengan memaksimalkan kemungkinan vektor-vektor kata yang disematkan di bawah kelas. Dalam dua dataset patokan klasifikasi standar, satu adalah keseimbangan dan yang lainnya adalah ketidakseimbangan, model kami mengungguli sistem mutakhir, pada akurasi dan skor F-1 rata-rata makro.

Penulis	Amin Hedayati Moghaddam, Moein Hedayati Moghaddam, dan Morteza Esfandyari
No. ISSN	22180648
Terbitan	Journal of Economics, Finance and Administrative Science
Tahun	2016
Judul	<i>Predicting Stock Market Index Using Artificial Neural Network</i>
Deskripsi	Dalam penelitian ini kemampuan jaringan syaraf tiruan (JST) dalam meramalkan nilai tukar harian NASDAQ. Beberapa umpan maju JST yang dilatih oleh algoritma propagasi balik telah dinilai. Metodologi yang digunakan dalam penelitian ini mempertimbangkan harga saham historis jangka pendek serta hari dalam seminggu sebagai input. Nilai tukar harian saham NASDAQ dari 28 Januari 2015 hingga 18 Juni 2015 digunakan untuk mengembangkan model yang kuat. 70 hari

	pertama (28 Januari hingga 7 Maret) dipilih sebagai set data pelatihan dan 29 hari terakhir digunakan untuk menguji kemampuan prediksi model. Jaringan untuk prediksi indeks NASDAQ untuk dua jenis dataset input (empat hari sebelumnya dan sembilan hari sebelumnya) dikembangkan dan divalidasi.
--	---

Penulis	Bobby Nazief, Mirna Adriani, Jelita Asian, Hugh E. Williams, dan S. M.M. Tahaghoghi
No. ISSN	14451336
Terbitan	Conferences in Research and Practice in Information Technology Series
Tahun	2005
Judul	<i>Stemming Indonesian</i>
Deskripsi	<i>Word Stemming</i> dapat diaplikasikan dalam pencarian teks, terjemahan mesin, ringkasan dokumen, dan klasifikasi teks. Dalam pencarian teks, ini memungkinkan pencarian untuk menemukan dokumen yang berisi semua kata dengan kata dasar yang dicari. Dalam bahasa Indonesia, <i>stemming</i> sangat penting: kata-kata memiliki awalan, sufix, infiks yang membuat pencocokan kata-kata yang terkait menjadi sulit. Dalam tulisan ini, kami menyelidiki kinerja algoritma <i>stemming</i> Bahasa Indonesia melalui studi pengguna. Hasil kami menunjukkan bahwa, dengan ketersediaan kamus yang masuk akal, algoritme Nazief dan Adriani yang tidak dipublikasikan dengan benar memuat sekitar 93% kemunculan kata ke kata dasar yang benar. Dengan perbaikan yang kami usulkan, ini hampir mencapai 95%. Kami menyimpulkan bahwa stemming untuk bahasa Indonesia harus dilakukan dengan menggunakan pendekatan Nazief dan Adriani kami yang telah dimodifikasi.

Penulis	Sutikno, Indriyati, S. S. Priyo, A. W. Helmie, W. Indra, B. Nurdin, dan Tri Wardati K.
No. ISSN	2086 – 4930
Terbitan	Jurnal Masyarakat Informatika
Tahun	2014

Judul	Aplikasi Prediksi Jumlah Penderita Penyakit Demam Berdarah <i>Degue</i> di Kota Semarang Menggunakan Jaringan Syaraf Tiruan <i>Backpropagation</i>
Deskripsi	IR (incidence rate) DBD Kota Semarang selalu lebih tinggi dari IR DBD Jawa Tengah dan IR DBD Nasional dari tahun 2006 sampai tahun 2012. Tahun 2012 IR DBD Kota Semarang adalah 70,9, tiga kali lebih tinggi dari IR DBD Jawa Tengah yaitu 19,29. Tindakan preventif dari pemerintah diharapkan dapat menekan peningkatan jumlah penderita. Dengan mengetahui peningkatan jumlah penderita, maka pemerintah dapat melakukan tindakan apa yang harus dilakukan untuk menekan penyebaran penyakit. Data jumlah penderita penyakit bersifat kontinu dan memiliki tren yang dinamis, maka metode yang dibutuhkan untuk melakukan prediksi adalah metode yang kompleks dan dapat mempelajari ketidakpastian dalam setiap periode yang dapat diakomodasi dengan algoritme Jaringan Syaraf Tiruan (JST). Salah satu algoritme JST yaitu Backpropagation. Jumlah penderita penyakit DBD per bulan dalam satu tahun digunakan sebagai masukan dengan jumlah penderita penyakit pada satu bulan ke depan sebagai keluaran. Arsitektur backpropagation menggunakan dua belas neuron layer input, satu layer tersembunyi dengan jumlah neuron yang dapat diubah-ubah dan satu neuron pada layer output. Pengujian menghasilkan MAPE terkecil, yaitu 4.48 pada alfa 0.7 dan jumlah neuron tersembunyi sebanyak 50 neuron dengan nilai akurasi 95.52 persen.

Penulis	Fadillah Z. Tala
No. ISSN	2443-1311
Terbitan	Jurnal Teknosains
Tahun	2017
Judul	<i>A Study of Stemming Effect on Information Retrieval in Bahasa Indonesia</i>
Deskripsi	Stemming adalah proses yang menyediakan pemetaan varian kata morfologis yang berbeda ke dalam basis / kata umum (batang). Proses ini juga dikenal sebagai penggabungan. Berdasarkan asumsi bahwa istilah yang memiliki batang yang sama biasanya akan memiliki makna yang sama, proses stemming banyak digunakan dalam Pengambilan Informasi sebagai cara untuk meningkatkan kinerja pengambilan. Selain

	<p>kemampuannya untuk meningkatkan kinerja pengambilan, proses stemming, yang dilakukan pada waktu pengindeksan, juga akan mengurangi ukuran file indeks. Berbagai algoritma stemming untuk bahasa Eropa telah diusulkan. Desain dari stemmer ini berkisar dari teknik yang paling sederhana, seperti menghilangkan sufiks dengan menggunakan daftar sufiks yang sering, hingga desain yang lebih rumit yang menggunakan struktur morfologis kata-kata dalam proses inferensi untuk memperoleh batang. Algoritma ini juga telah dievaluasi untuk menguji pengaruhnya terhadap kinerja pengambilan.</p>
--	--



BAB 3

ANALISIS MASALAH DAN PERANCANGAN SOLUSI

3.1 Analisa Masalah

Seiring dengan berkembangnya teknologi dan informasi, memudahkan masyarakat untuk saling berinteraksi, dan bertukar informasi melalui media sosial, seperti Twitter. Hal ini sangat berperan penting dalam persebaran informasi secara luas dan cepat, mengingat besarnya jumlah pengguna Twitter aktif di Indonesia, akan sangat membantu jika masarakat dapat turut aktif memberikan dukungan berupa informasi kejadian bencana melalui media sosial Twitter. Dengan begitu pihak pemerintah dapat dengan cepat mengambil tindakan dan pertolongan kepada korban di lokasi bencana.

Tercatat berdasarkan data dari BNPB, dalam kurun waktu 10 tahun terakhir, telah terjadi sebanyak 4.971 kasus tanah longsor di Indonesia, dengan total korban terdampak dan menungsi sebanyak 216.028 orang. Oleh karenanya, upaya penanggulangan yang cepat, serta peringatan terhadap gejala bencana tanah longsor sangatlah perlu untuk dikembangkan, guna mengurangi angka korban jiwa yang tinggi akibat bencana tersebut.

Peringatan dan laporan mengenai kejadian bencana yang saat ini dilakukan oleh BNPB adalah dengan mengandalkan laporan-laporan dari pihak BPBD (Badan Penanggulangan Bencana Daerah) yang selanjutnya akan diteruskan ke pusat untuk dilakukan tindakan. Hal ini tentunya menjadi kurang efektif mengingat luasnya wilayah negara Indonesia, serta terbatasnya instrumen dan sumber daya manusia yang dimiliki oleh BPBD.

Melihat upaya yang dilakukan saat ini, masih memungkinkan untuk dilakukan sebuah terobosan di mana masyarakat dapat secara langsung ikut serta dalam melaporkan kejadian melalui media sosial.

Twitter menjadi salah satu media sosial yang paling banyak digunakan oleh masyarakat. Dengan adanya media sosial Twitter, informasi dapat dengan cepat tersebar, termasuk di dalamnya tweet – tweet mengenai informasi bencana tanah longsor. Tercatat ada sekitar 5600 tweet dengan kata kunci “longsor” yang didapat dari tahun 2016 s/d 2019.

Dengan jumlah *dataset* yang besar, dibutuhkan sebuah sistem yang dapat mengklasifikasi secara otomatis *tweet* yang termasuk ke dalam kelas peringatan bencana atau bukan. Berdasarkan penjelasan diatas, penulis mencoba mengimplementasikan algoritme *Artificial Neural Network* (ANN), yang akan mengklasifikasi secara otomatis data-data tweet dengan kata kunci “longsor” yang didapat dari Twitter, guna mengindikasi tweet yang memang mengandung peringatan atau laporan kejadian tanah longsor yang sedang terjadi. Pengklasifikasian tersebut berdasarkan bobot tiap kata yang terdapat dalam setiap tweet .

3.2 Spesifikasi *Database*

Berikut ini merupakan struktur tabel yang digunakan dalam pembuatan *database* untuk aplikasi ini. Database yang digunakan berkaitan dengan aplikasi ini hanya berfungsi untuk menyimpan kumpulan *tweet* sebagai data yang digunakan dalam sistem klasifikasi teks.

1) Tabel Dataset

Nama Tabel : data_tweet
 Media : Harddisk
 Primary Key : id

Tabel 3.1 Dataset

No	Field	Type	Length	Keterangan
1	id	bigint	20	Primary key data tweet
2	username	varchar(255)	255	Akun pengguna dari tweet
3	name	varchar(255)	255	Nama pengguna dari tweet
4	content	text	-	Isi tweet
5	score_0	double	-	Nilai klasifikasi untuk kelas Non-warning
6	score_1	double	-	Nilai klasifikasi untuk kelas Warning
7	class	char	1	Kelas hasil klasifikasi
8	created_at	timestamp	-	

3.3 Penyelesaian Masalah

Berdasarkan uraian masalah di atas, maka penulis akan mengimplementasikan algoritme klasifikasi teks, untuk menentukan apakah ada laporan kejadian, maupun peringatan tanah longsor oleh pengguna Twitter di Indonesia. Algoritme yang digunakan adalah *Artificial Neural Network* (Jaringan Saraf Tiruan). *Artificial Neural Network* bekerja dengan mengatur bobot dari setiap *input* pada *input layer* (pada kasus klasifikasi teks, *input* yang digunakan berupa kata). Melalui proses *training*, setiap kata akan memiliki skor, yang mana melalui perulangan-perulangan, algoritme akan menyesuaikan skor dari setiap *input*, relatif mendekati dengan ukuran kelas.

Langkah pertama, sistem akan menganalisis seluruh data *tweet* yang dijadikan *dataset*. Kemudian memilih sebanyak 300 kata yang paling

dominan muncul di setiap *tweet* yang telah disaring berdasarkan kata kunci “longsor”, untuk dijadikan sebagai *input* pada *input layer*.

Selanjutnya, nilai *bobot* dan jumlah *hidden layer* diinisialisasikan dengan nilai acak, untuk kemudian melalui proses *training* bisa ditemukan angka yang tepat dan paling optimal dan mendekati hasil klasifikasi yang diharapkan.

Terakhir, pada tahap pengujian, komposisi bobot yang sudah dirumuskan melalui proses *training*, diuji coba terhadap data *tweet* untuk mengetahui apakah data *tweet* tersebut mendapat klasifikasi kelas yang sesuai dengan data *tweet* yang telah dilabeli manual sebelumnya.

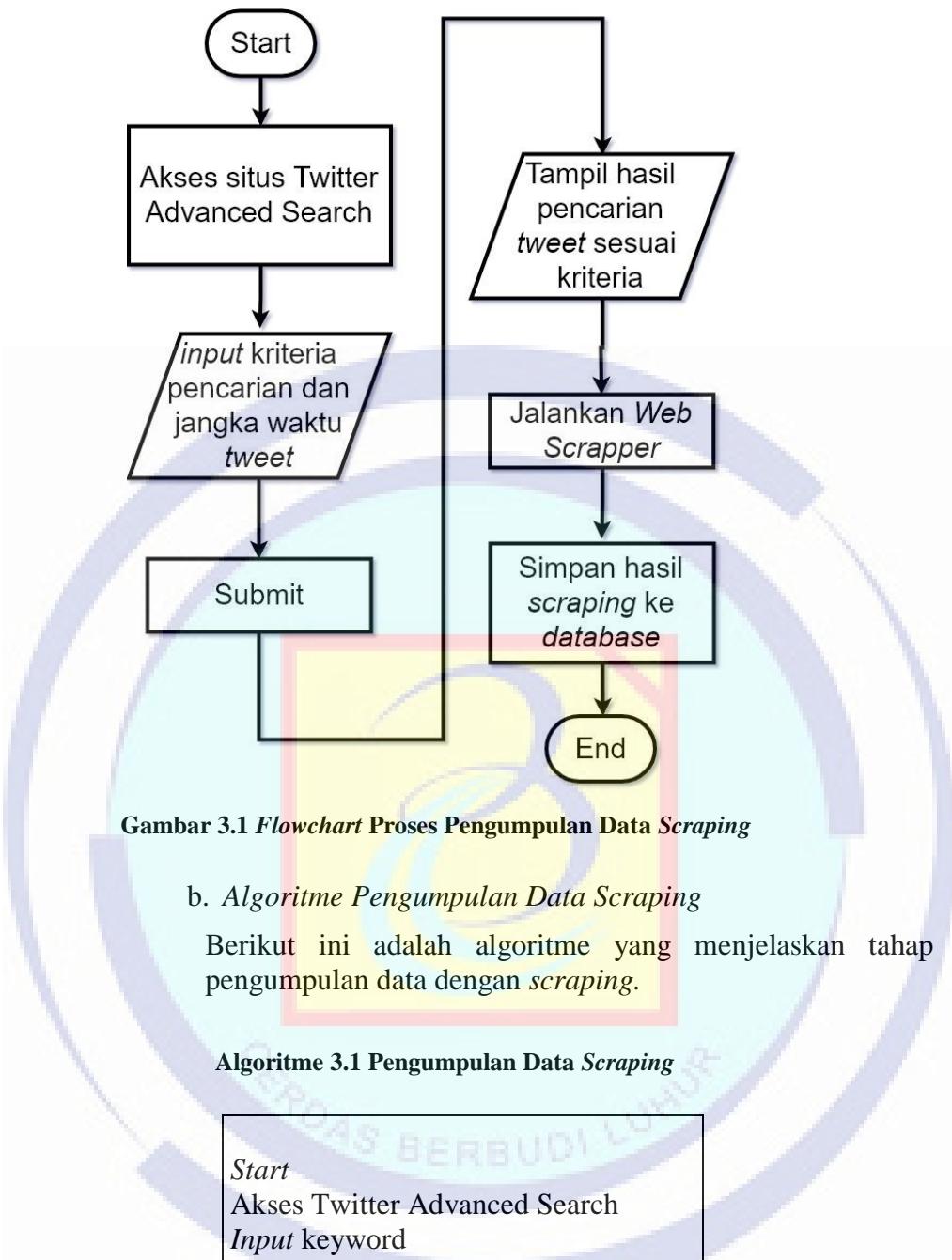
3.4 Algoritme dan *Flowchart*

Setelah dibuatnya rancangan layar selanjutnya untuk memperjelas serangkaian alur program yang telah dibuat dari rancangan layar, maka dibuatlah *flowchart* dan algoritme dalam penggunaan aplikasi agar pengguna bisa menggunakannya dengan benar. *Flowchart* sendiri digunakan untuk mengarahkan alur proses program. Sedangkan algoritma digunakan untuk memperjelas dari penjelasan *flowchart*. Berikut adalah *flowchart* dan algoritme dari setiap proses yang ada pada aplikasi.

3.4.1 Pengumpulan Data Scraping

a. *Flowchart* Pengumpulan Data Scraping

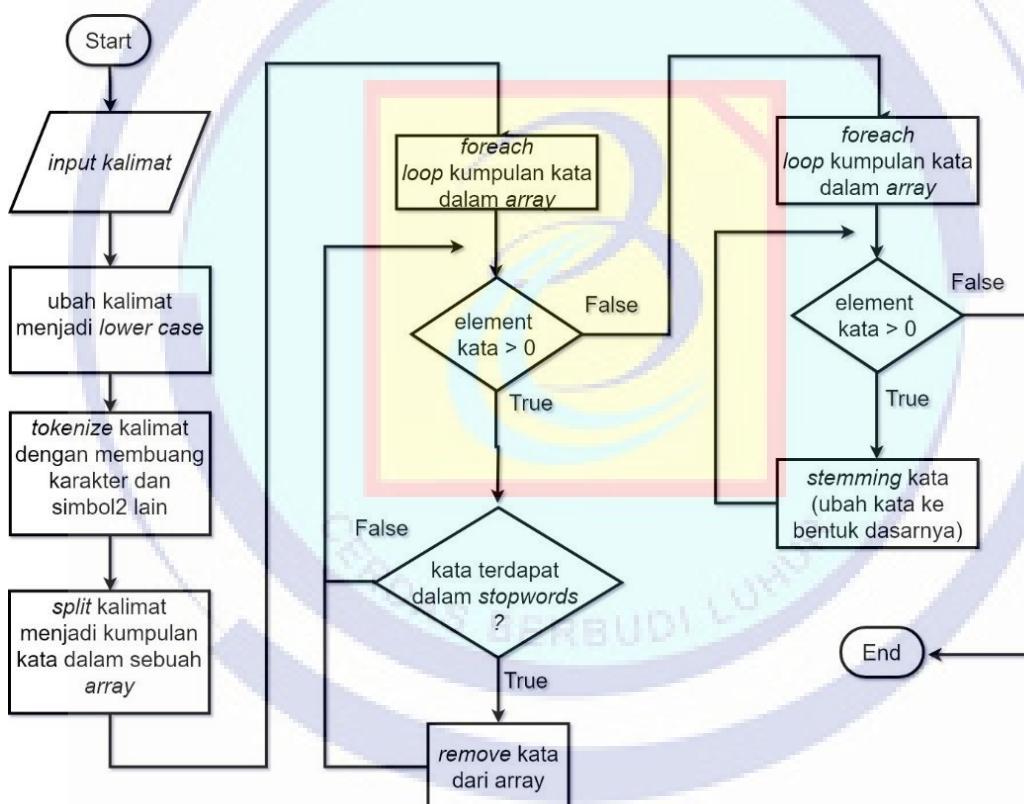
Flowchart ini menjelaskan mengenai teknik pengumpulan data yang digunakan untuk mendapatkan *dataset* berupa *tweet*. Penulis menggunakan perangkat lunak Web Scraper yang merupakan aplikasi tambahan (*extension*) dari aplikasi Chrome Browser. Web Scraper dapat dengan otomatis mendapatkan kumpulan *tweet* hasil dari pencarian pada laman *web* Twitter Advanced Search. Twitter Advanced Search memberikan fasilitas bagi pengguna secara terbuka, melakukan pencarian dengan banyak kriteria. Sebagaimana terlihat pada Gambar 3.1.



3.4.2 Preprocessing

a. Flowchart Preprocessing

Flowchart ini menjelaskan mengenai tahapan *preprocessing*, yaitu tahap untuk membersihkan isi *dataset* menjadi format yang lebih mudah untuk diolah dengan algoritme *backpropagation artificial neural network*. Tahap ini terdiri dari *case folding* (pengubahan kalimat menjadi *lower case*), *tokenizing* (membuang karakter dan simbol-simbol tidak terpakai), *split* kalimat menjadi kumpulan kata, penyaringan kata-kata *stopwords*, dan *stemming* (mengubah kata ke bentuk kata dasarnya). *Flowchart* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Flowchart Preprocessing

b. Algoritme Preprocessing

Berikut ini adalah algoritme yang menjelaskan tahap *preprocessing*.

Algoritme 3.2 Algoritme Preprocessing

```
Start
Input kalimat
kalimat.lower() #ubah kalimat ke lower case
kalimat.sub('[^A-Za-z0-9# ]+', " ") #proses tokenize
kalimat.split() #ubah kalimat menjadi kumpulan kata
foreach kata in kalimat:
    if kata in_array(stopwords):
        kalimat.remove(kata)

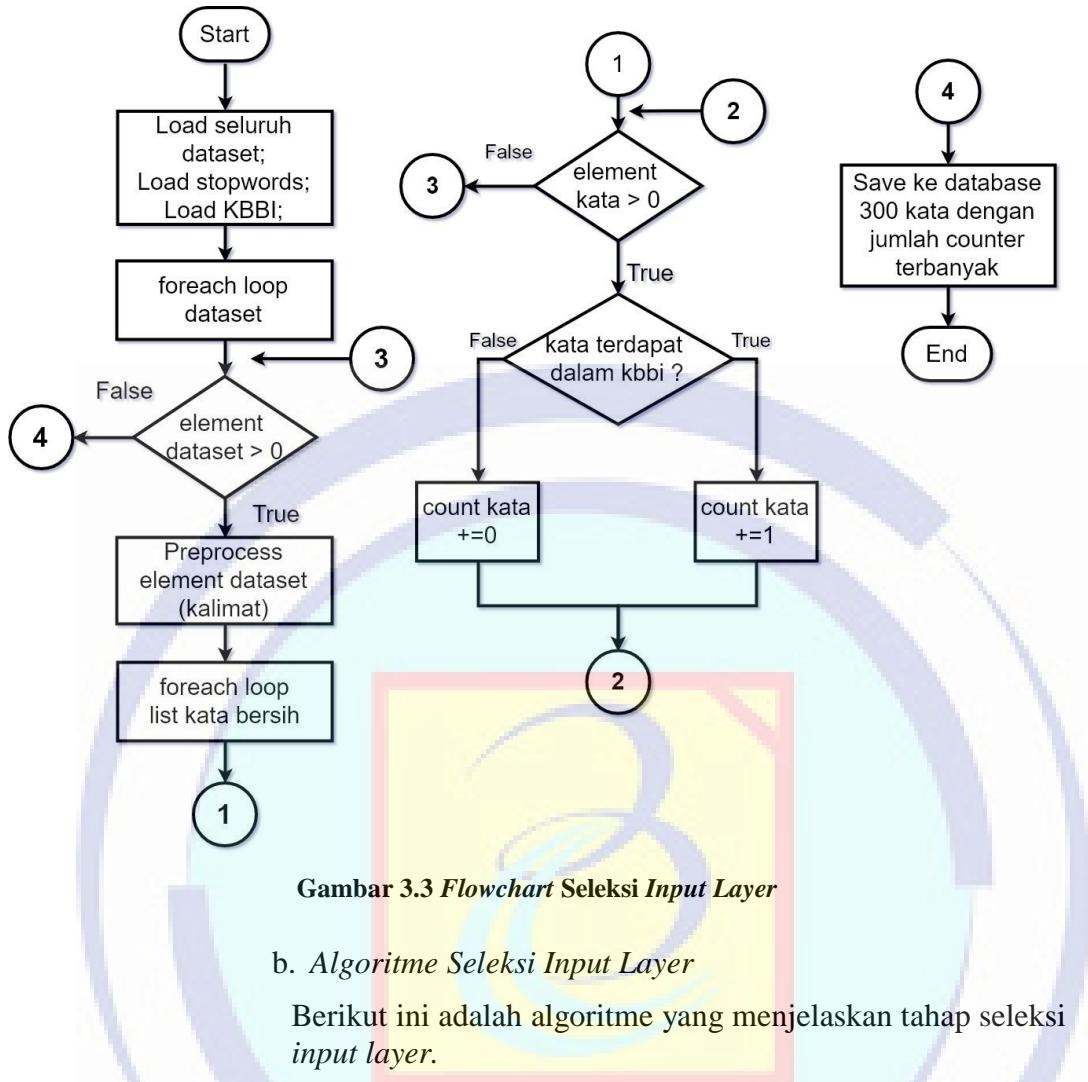
kata_stemmed = [] #variabel penampung kata yg bersih
foreach kata in kalimat:
    kata_stemmed.append( kata.stem() )

return kata_stemmed;
End
```

3.4.3 Seleksi Input Layer

a. Flowchart Seleksi Input Layer

Flowchart ini menjelaskan mengenai tahapan dalam seleksi kata-kata yang paling dominan muncul dari keseluruhan *dataset* yang akan melalui proses *training*. Hal ini dilakukan untuk mengoptimalkan akurasi dan efisiensi klasifikasi. Tahap ini bekerja dengan menghitung jumlah kemunculan kata-kata yang sama dari setiap *tweet* yang dijadikan *dataset*. Kemudian dipilih 200 kata yang paling dominan muncul sebagai masukan pada *input layer*. Seperti terlihat pada Gambar 3.3.



```

Start
Load "twitter_dataset"
Load "stopwords"
Load "KBBI"

foreach tweet in twitter_dataset:
    clean_tweet = Preprocess(tweet)
    foreach word in clean_tweet:
        if word in_array(KBBI):
            KBBI[word] += 1

sort_desc(KBBI)
for i in range(300):
    insert_database(KBBI[i]);
    
```

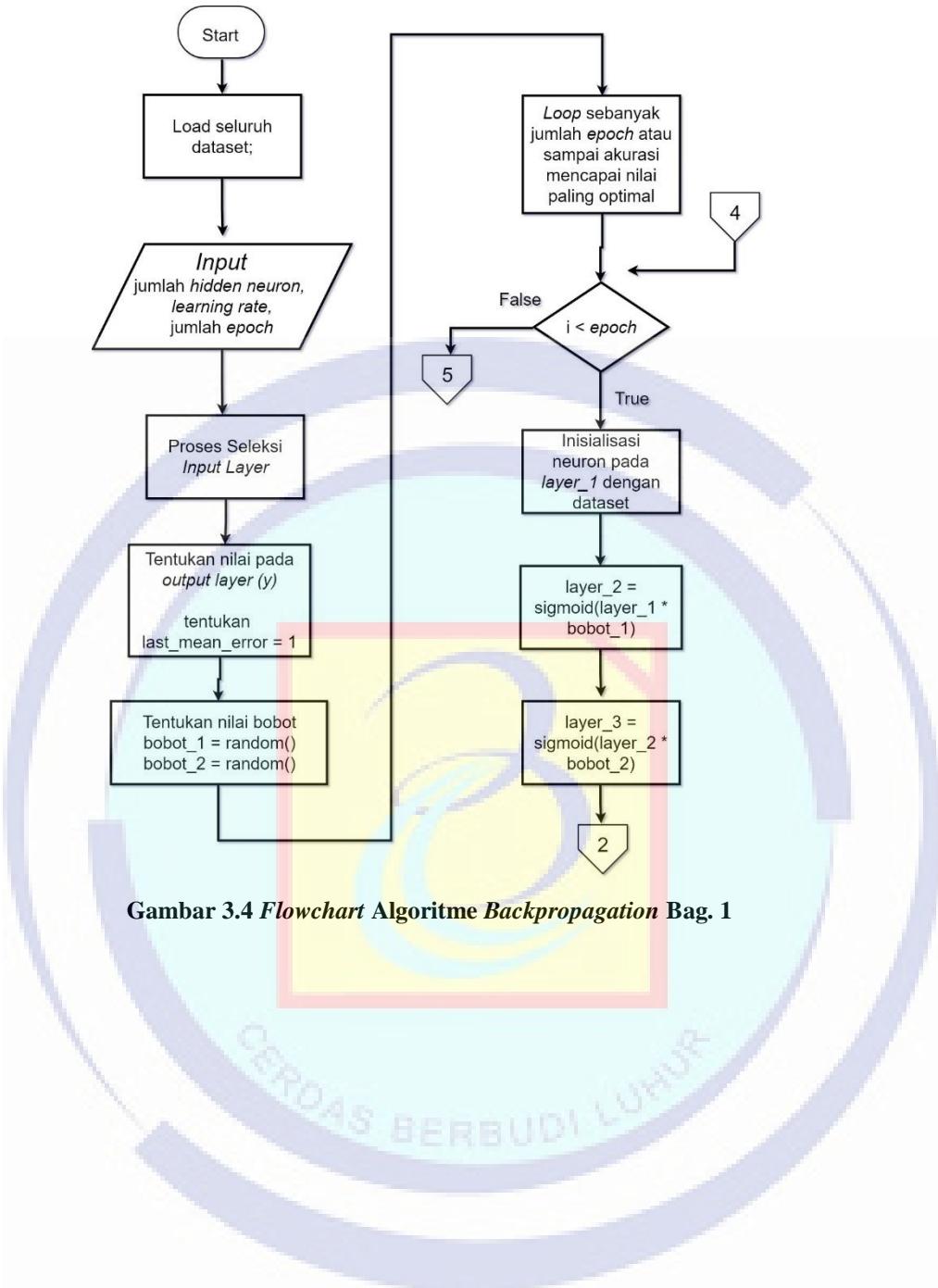
End

Algoritme 3.3 Seleksi *Input Layer*

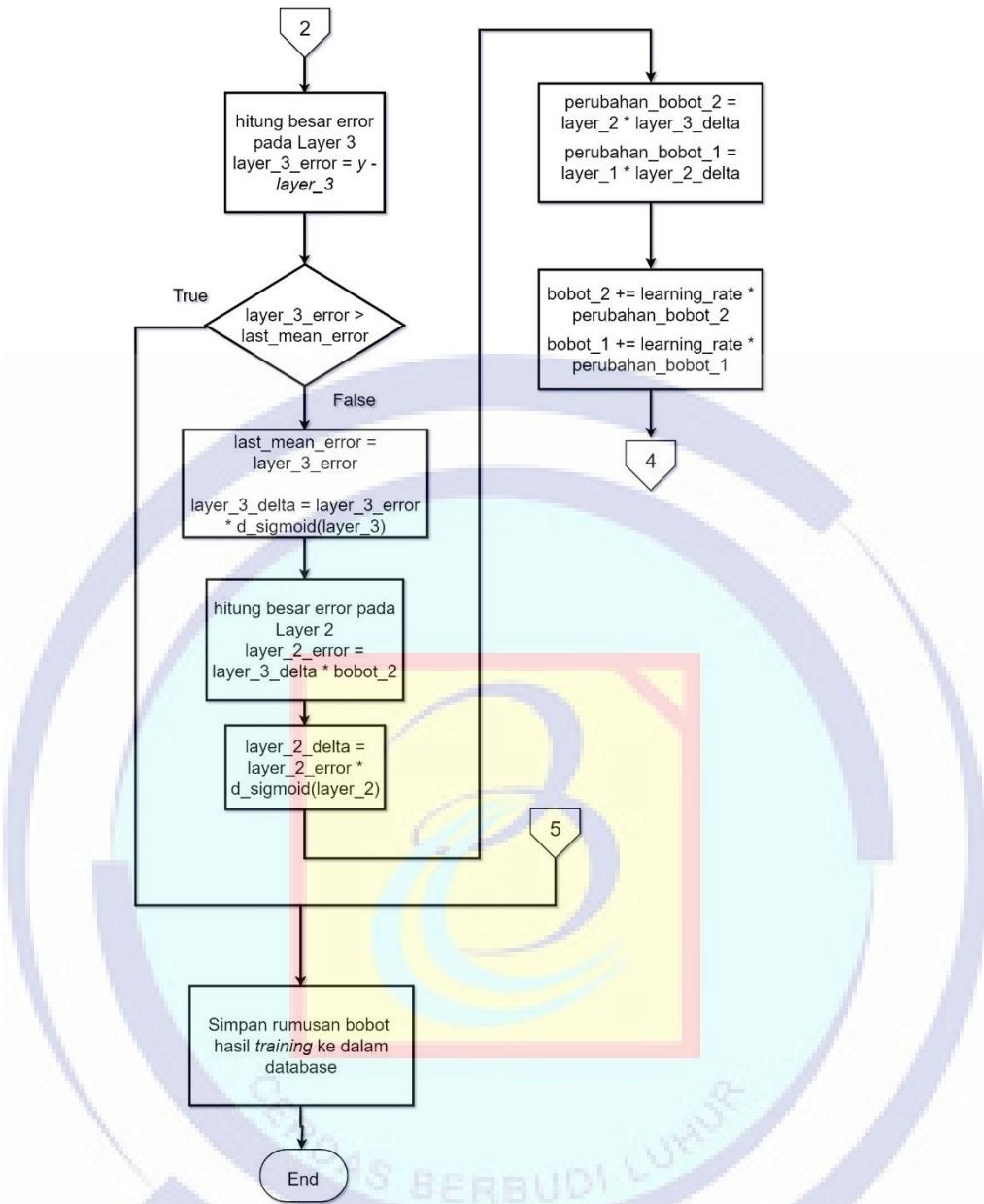
3.4.4 *Training Data (Backpropagation)*

a. *Flowchart Training Data (Backpropagation)*

Flowchart ini menjelaskan mengenai algoritme *backpropagation* yang digunakan pada tahapan *training*. Inti dari algoritme ini adalah untuk menemukan bobot yang paling optimal dengan data *training*, sehingga mendekati dengan kelas yang telah ditentukan sebelumnya. Hasil dari *training* data menggunakan algoritme *backpropagation* adalah rumusan bobot-bobot untuk kemudian digunakan pada proses prediksi. Alur dari proses algortime *backpropagation* dapat dilihat pada Gambar 3.4 dan Gambar 3.5.



Gambar 3.4 Flowchart Algoritme Backpropagation Bag. 1



Gambar 3.5 Flowchart Algoritme *Backpropagation* Bag. 2

b. Algoritme Training Data (*Backpropagation*)

Berikut ini adalah algoritme yang menjelaskan tahap *training* dengan metode *backpropagation*.

Algoritme 3.4 *Backpropagation*



```
Load expected_output_class
Input hidden_neuron
Input learning_rate
Input epoch

Seleksi_Input_Layer()

#inisialisasi y dengan output yg diinginkan
y = expected_output_class

last_mean_error = 1
bobot_1 = random()
bobot_2 = random()

for i in range(epoch):
    layer_1 = dataset #input layer
    layer_2 = sigmoid(layer_1 * bobot_1) #hidden layer
    layer_3 = sigmoid(layer_2 * bobot_2) #output layer

    layer_3_error = y - layer_3
    if layer_3_error > last_mean_error:
        break;

    last_mean_error = layer_3_error
    layer_3_delta = layer_3_error * d_sigmoid(layer_3)
    layer_2_error = layer_3_delta * bobot_2
    layer_2_delta = layer_2_error * d_sigmoid(layer_2)

    perubahan_bobot_2 = layer_2 * layer_3_delta
    perubahan_bobot_1 = layer_1 * layer_2_delta

    bobot_2 += learning_rate * perubahan_bobot_2
    bobot_1 += learning_rate * perubahan_bobot_1

save_to_database (bobot_1, bobot_2)

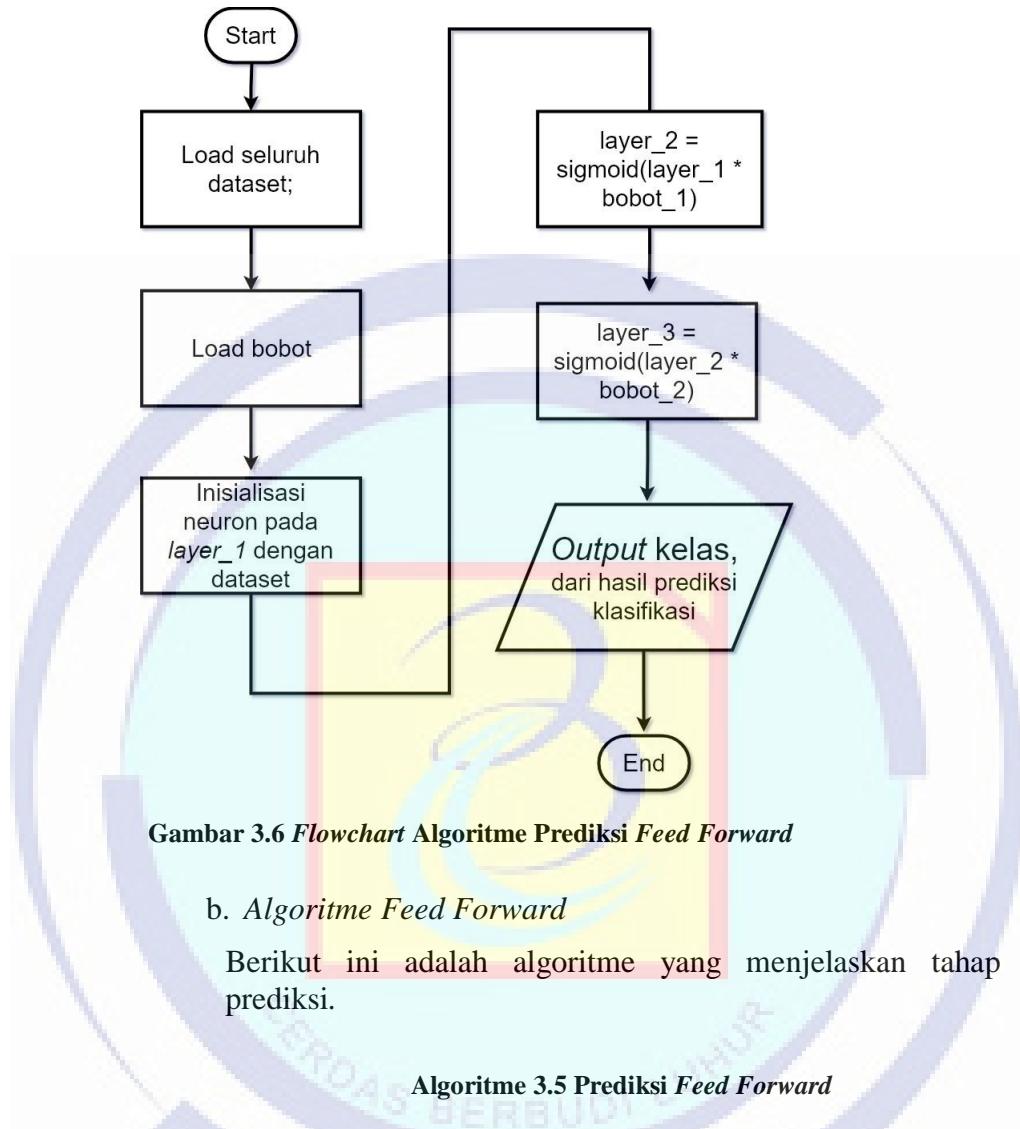
End
```

3.4.5 Feed Forward (Prediksi)

a. Flowchart Feed Forward

Flowchart ini menjelaskan mengenai proses prediksi sebuah dataset, menggunakan rumusan bobot yang telah didapatkan dari hasil proses *training* sebelumnya. Hasil akhir dari proses ini menunjukan kelas apa yang paling sesuai

dengan *dataset* yang diberikan tersebut. *Flowchart* tahap *feed forward*, dapat dilihat pada Gambar 3.6.



```

Start
Load dataset
Load bobot

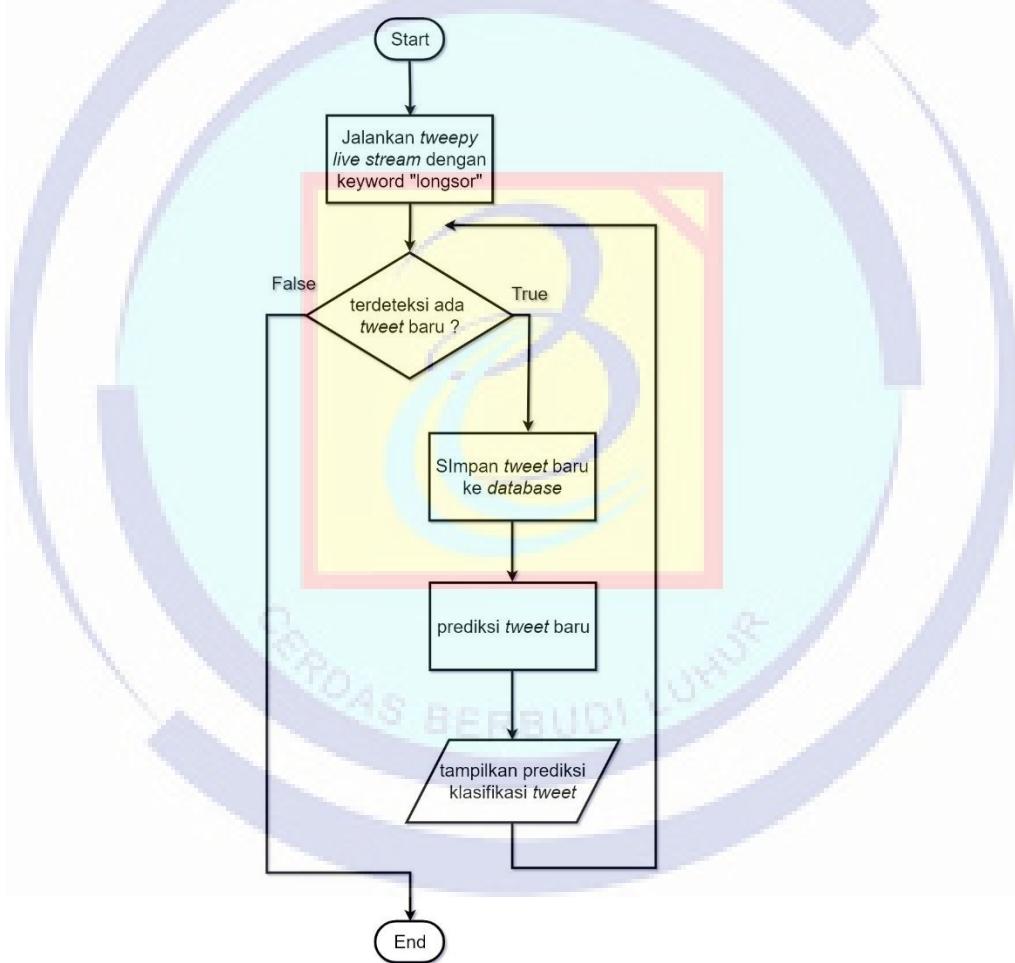
layer_1 = dataset #input layer
layer_2 = sigmoid(layer_1 * bobot_1) #hidden layer
layer_3 = sigmoid(layer_2 * bobot_2) #output layer

print (layer_3) #tampilkan hasil prediksi pada layer_3
End
  
```

3.4.6 Menu Monitoring dan Streaming Live Data

a. Flowchart Menu Monitoring dan Streaming Live Data

Flowchart ini menjelaskan bagaimana interaksi yang terjadi pada menu *monitoring* sekaligus menu utama dari aplikasi yang dibuat. Penulis menggunakan *library* terbuka bernama *tweepy* untuk melakukan *request live streaming* data-data *tweet* terbaru dari Twitter, dengan kata kunci “longsor”. Jika terdeteksi adanya *tweet* baru dengan aturan penyaringan tadi, maka aplikasi akan otomatis mengunduh dan kemudian memprediksi *tweet* baru tersebut, yang kemudian langsung ditampilkan pada tabel di menu *monitoring*. Seperti terlihat pada Gambar 3.7.



Gambar 3.7 Flowchart Menu Monitoring dan Live Streaming tweet

b. Algoritme Menu Monitoring dan Streaming Live Data

Berikut ini adalah algoritme yang menjelaskan interaksi pengguna pada menu *monitoring* dan *live streaming data*.

Algoritme 3.6 Menu *Monitoring dan Live Stream*

```
Start  
tweepy.stream("longsor")  
  
if tweepy.result() != null:  
    new_tweet = tweepy.result()  
    save_database(new_tweet)  
    tweet_class = predict(new_tweet)  
    print(tweet_class)
```

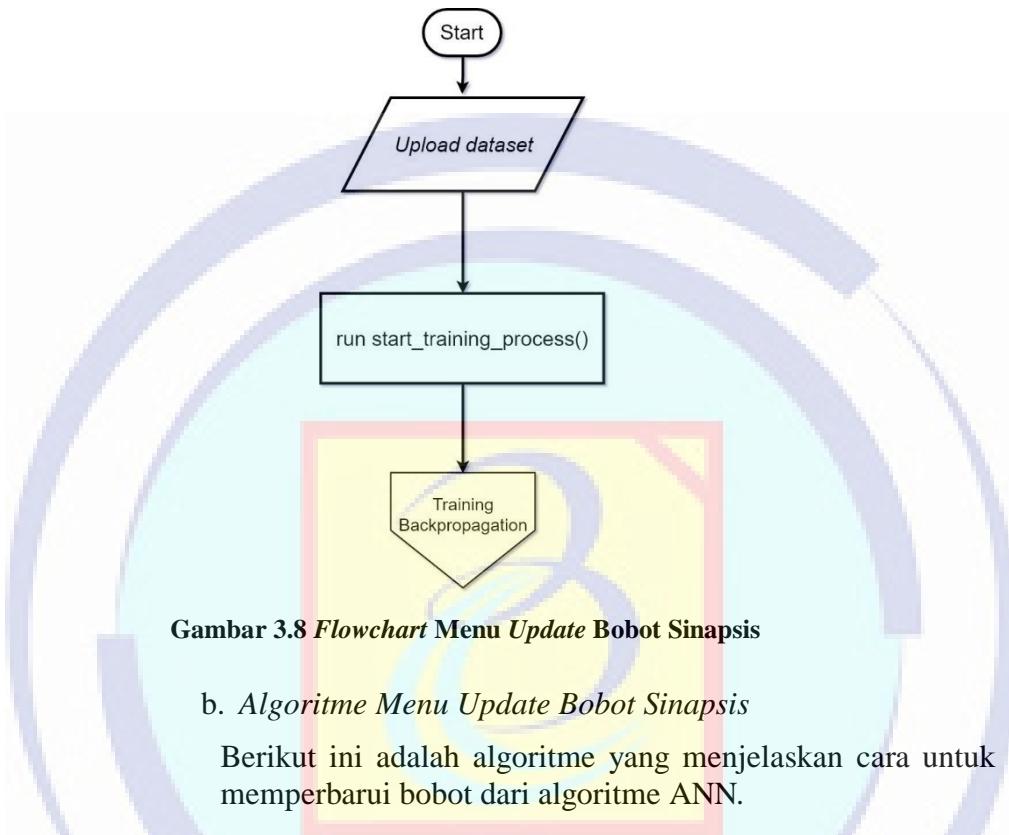
End



3.4.7 Menu Update Bobot Sinapsis

a. Flowchart Menu Update Bobot Sinapsis

Flowchart ini menjelaskan cara interaksi pengguna pada menu pengaturan untuk memperbarui bobot pada *artificial neural network*. Seperti yang terlihat pada Gambar 3.8.



b. Algoritme Menu Update Bobot Sinapsis

Berikut ini adalah algoritme yang menjelaskan cara untuk memperbarui bobot dari algoritme ANN.

Algoritme 3.7 Menu Perbarui Bobot

```
file_dir = "dataset_training.csv"  
nn.start_training_process(data_source = file_dir)  
weight saved in "synapses.json"  
End
```

3.5 Simulasi Alur dan Perhitungan Algoritme

3.5.1 Menentukan Input Layer

Penentuan *term* (kata) yang akan digunakan sebagai *input neuron* adalah dengan menghitung *term frequency* dari seluruh dataset yang digunakan. Penulis membatasi jumlah *input neuron* menjadi 200 kata dengan frekuensi kemunculan tertinggi dalam keseluruhan dataset. Daftar tabel kata yang digunakan dalam penelitian, dapat dilihat pada Tabel 3.2.



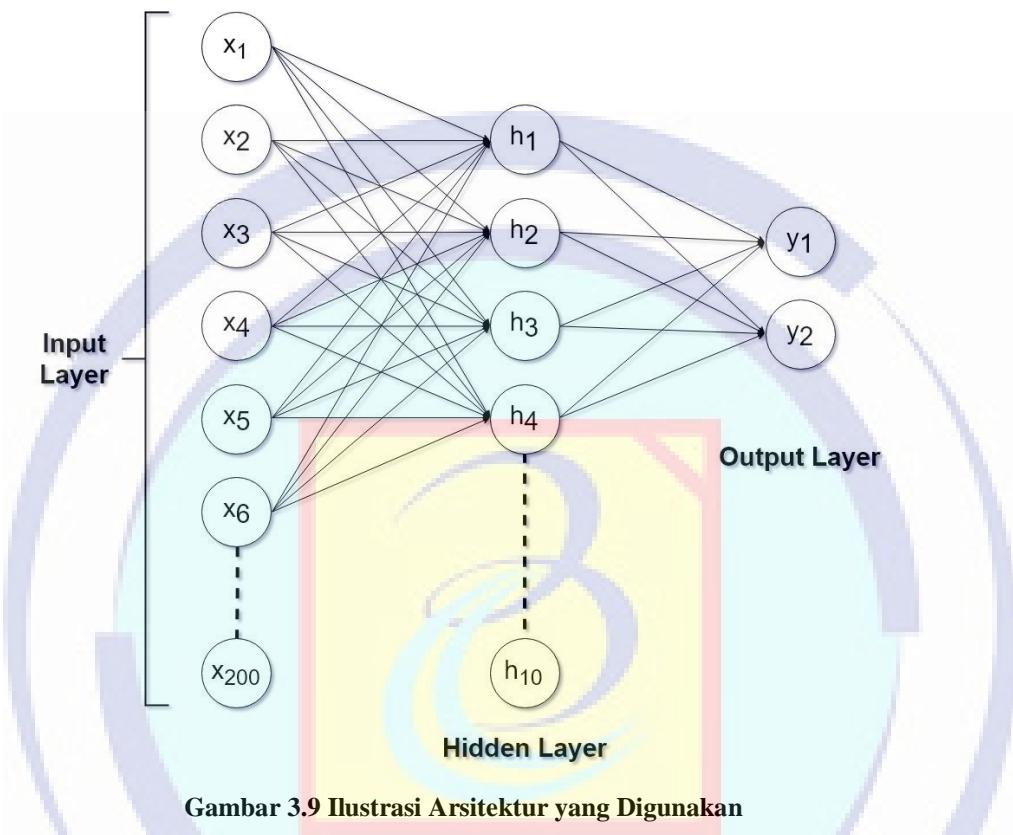
Tabel 3.2 Daftar Term pada Input Layer

No.	Term	No.	Term	No.	Term	No.	Term	No.	Term
1	longsor	41	ada	81	sungai	121	total	161	terima
2	korban	42	barat	82	alat	122	kali	162	cita
3	banjir	43	ini	83	tinjau	123	sumbar	163	kemarin
4	tanah	44	timpa	84	dusun	124	mari	164	bahu
5	bencana	45	jiwa	85	pagi	125	bal	165	gerak
6	jalan	46	bogor	86	tambang	126	dalam	166	tidak
7	desa	47	deras	87	berat	127	giat	167	tugas
8	orang	48	selatan	88	malam	128	lebat	168	minggu
9	bantu	49	selamat	89	kondisi	129	alami	169	jenazah
10	hujan	50	via	90	jam	130	anak	170	bandara
11	rumah	51	tebing	91	peduli	131	bandung	171	beri
12	akibat	52	hilang	92	jembatan	132	tanggap	172	guyur
13	warga	53	rusak	93	dunia	133	langsung	173	hotel
14	timbun	54	akses	94	kerja	134	musim	174	siap
15	rawan	55	wilayah	95	bukit	135	hutan	175	masih
16	jalur	56	masyarakat	96	keluarga	136	puluh	176	butuh
17	tewas	57	ungsi	97	hari	137	lapor	177	serah
18	jadi	58	putus	98	hasil	138	pasang	178	bawah
19	di	59	bersih	99	timur	139	ingat	179	salah
20	tutup	60	duka	100	bangun	140	foto	180	februari
21	evakuasi	61	terjang	101	buka	141	cuaca	181	ka
22	lokasi	62	titik	102	saudara	142	kuat	182	jumat
23	daerah	63	tsunami	103	bisa	143	bagi	183	lindung
24	waspadा	64	lintas	104	kampung	144	turun	184	terjun
25	temu	65	air	105	mohon	145	sore	185	salem
26	tinggal	66	ancam	106	arah	146	puting	186	lereng
27	cari	67	kena	107	ruas	147	belitung	187	dana
28	kabupaten	68	alam	108	salur	148	lanjut	188	badan
29	dampak	69	material	109	susul	149	tingkat	189	curah
30	tim	70	indonesia	110	ya	150	meter	190	tahan
31	camat	71	doa	111	yang	151	raya	191	lombok
32	gempa	72	potensi	112	allah	152	beberapa	192	mudik
33	gunung	73	laku	113	pohon	153	bumi	193	donasi
34	kota	74	sar	114	kendara	154	atas	194	hubung
35	landa	75	info	115	siaga	155	dari	195	tumbang
36	puncak	76	baik	116	batu	156	padang	196	bakti
37	tangan	77	bandang	117	proses	157	kait	197	pasir
38	sebab	78	darurat	118	gabung	158	provinsi	198	tambah
39	musibah	79	video	119	utara	159	luka	199	berita
40	moga	80	kawasan	120	cepat	160	angin	200	upaya

3.5.2 Proses Training

Proses pelatihan *dataset* diawali dengan penentuan arsitektur *Artificial Neural Network* yang akan digunakan. Pada penelitian ini digunakan *multi-layer* yang terdiri dari *input layer*, 1 *hidden layer*,

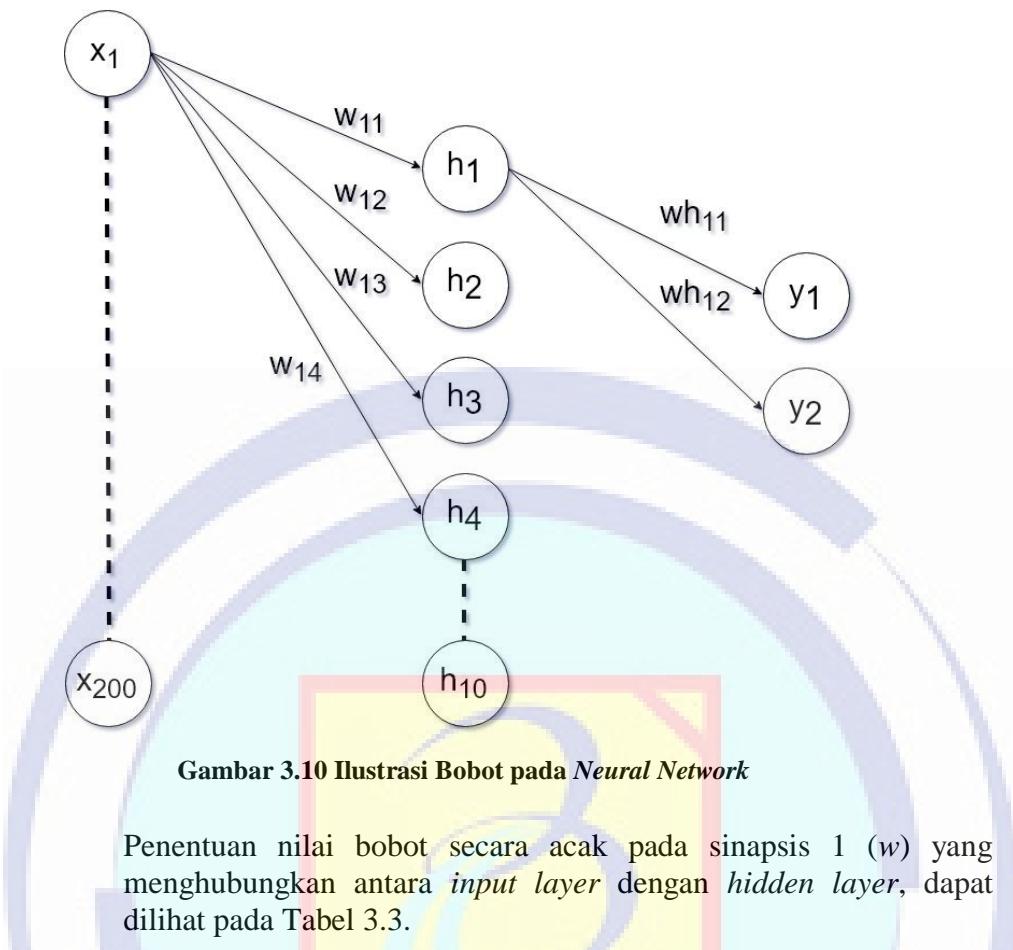
dan *output layer*. *Input layer* terdiri dari 200 *neuron* yang merupakan representasi biner dari daftar *term* (kata) yang digunakan. *Hidden layer* terdiri dari 10 *neuron*, dan *output layer* terdiri atas 2 *neuron* yang mewakili kelas *warning* (dengan nilai 1) dan *non-warning* (dengan nilai 0). Ilustrasi dari arsitektur yang digunakan, dapat dilihat pada Gambar 3.9.



Gambar 3.9 Ilustrasi Arsitektur yang Digunakan

Selanjutnya untuk memudahkan ilustrasi penghitungan dalam proses algoritme *Backpropagation Artificial Neural Network*, penulis hanya mengambil sampel sebanyak 6 *neuron* pertama dari *term input layer* pada Tabel 3.2, dan menggunakan 3 buah *neuron* pada *hidden layer*.

Tahap awal dari proses *backpropagation* adalah dengan menentukan bobot awal secara acak. Bobot dapat diilustrasikan sebagai penghubung antar *neuron* pada tiap *layer*, atau bisa juga disebut sinapsis. Ilustrasi dari bobot neuron dan sinapsis antar *layer*, dapat dilihat pada Gambar 3.10.



Tabel 3.3 Daftar Bobot Sinapsis 1 (w)

	w_1	w_2	w_3
longsor	0,7	-0,8	0,2
korban	0,1	-0,5	0,3
banjir	0,1	0,5	-0,4
tanah	0,7	-0,7	0,4
bencana	0,5	-0,2	0,3
jalan	0,6	-0,2	0,7

Sementara untuk penentuan nilai bobot secara acak pada sinapsis 2 (wh) yang menghubungkan antara *hidden layer* dengan *output layer*, dapat dilihat pada tabel 3.4.

Tabel 3.4 Daftar Bobot Sinapsis 2 (wh)

	wh_1	wh_2
h_1	-0.1	0.4
h_2	0.5	-0.1
h_3	0.5	0.9

Selanjutnya, untuk mensimulasikan perhitungan algoritme *backpropagation* ini, penulis menggunakan satu data (*tweet*) sebagai berikut :

Kapolda Sulsel Terjun Langsung Pimpin Evakuasi **Korban Bencana Longsor** di Gowa Kapolda Sulawesi Selatan (Sulsel), Irjen Pol. Drs. Hamidin memimpin langsung proses evakuasi **korban bencana longsor** di Desa Pattalikang, Kecamatan Manuju, Kabupaten Gowa, Sulawesi Selatan. pic.twitter.com/OmTAGGz8MC

Kelas = Warning
 $y = [0, 1]$

Proses awal yang dilakukan dalam *training* menggunakan *backpropagation neural network*, adalah tahap *feed forward* untuk menghitung nilai pada *hidden layer*, dimana untuk menghitung nilai setiap *neuron* pada *layer-layer* yang ada menggunakan formula sebagai berikut :

$$h_j = f \left(\sum_{i=1}^n x_i w_{ij} \right) \dots \dots \dots \quad (18)$$

Dengan fungsi $f(x)$ yang digunakan adalah fungsi aktivasi sigmoid sebagaimana tertera pada rumus (1).

Hasil perhitungan *neuron* pada *hidden layer*, dapat dilihat pada Tabel 3.5.

Tabel 3.5 Perhitungan *Feed Forward* pada *Hidden Layer*

	x_i	$x_i w_1$	$x_i w_2$	$x_i w_3$
longsor (x_1)	1	0,7	-0,8	0,2
korban (x_2)	1	0,1	-0,5	0,3
banjir (x_3)	0	0	0	0
tanah (x_4)	0	0	0	0
bencana (x_5)	1	0,5	-0,2	0,3
jalan (x_6)	0	0	0	0
	$\sum x_i w_{ij}$	1,3	-1,5	0,8
	$f(x)$	0.78583498	0.18242552	0.68997448
		h₁	h₂	h₃

Perhitungan yang sama juga kita lakukan untuk mendapatkan nilai *neuron* pada *output layer*, maka hasil perhitungan dapat dilihat pada Tabel 3.6.

Tabel 3.6 Perhitungan *Feed Forward* pada *Output Layer*

	h_i	$h_i w h_1$	$h_i w h_2$
h₁	0.78583498	-0.078583498	0.314333992
h₂	0.18242552	0.09121276	-0.018242552
h₃	0.68997448	0.34498724	0.620977032
	$\sum h_i w h_{ij}$	0.3576165	0.91706847
	$f(x)$	0.58846333	0.71444441
		y₁	y₂

Setelah nilai *feed-forward* dari *output layer* ditemukan, tahap selanjutnya adalah memulai proses *backpropagation*. Diawali dengan menghitung nilai *error* dari *layer* terakhir (*output layer*), dengan cara menghitung selisih antara nilai *neuron* dari *output layer* dengan nilai *neuron* dari kelas yang telah ditentukan secara manual, seperti yang terlihat pada Tabel 3.7.

Tabel 3.7 Nilai Error pada Output Layer

		Label Manual (t)	Output Layer (Y)	Output Layer Error (selisih)
Non-warning	y1	0	0,58846333	-0,58846333
Warning	y2	1	0,71444441	0,28555559

Selanjutnya adalah menghitung nilai *delta* (perubahan) dari *output layer*, melalui formula sebagai berikut :

$$\delta y_k = Y_{err_k} * f'(Y_k) \dots \dots \dots (20)$$

Dengan formula turunan (*derivative*) dari fungsi aktivasi *sigmoid* sebagaimana tertera pada rumus (2), maka akan dihasilkan perhitungan seperti pada Tabel 3.8.

Tabel 3.8 Nilai Delta (Perubahan) pada Output Layer

		Output Layer Error (selisih)	δy_k
Non-warning	y1	-0.58846333	-0.58846333
Warning	y2	0.28555559	0.28555559

Selanjutnya adalah meneruskan perhitungan *backpropagation* dengan menghitung nilai kesalahan (*error*) dari *hidden layer*, melalui formula berikut :

$$h_{err_k} = \delta y_k * wh_k \dots \dots \dots (21)$$

Maka didapatkan hasil perhitungan sebagaimana tertera pada Tabel 3.9.

Tabel 3.9 Nilai Error pada Hidden Layer

	δy	$\delta y_j * wh_{1j}$	$\delta y_j * wh_{2j}$	$\delta y_j * wh_{3j}$	
δy_1	-0,58846333	0,058846333	-0,294231665	0,294231665	-
δy_2	0,28555559	0,114222236	-0,028555559	0,257000031	+
	$\sum \delta y_j * wh_{ij}$	0,173068569	-0,322787224	-0,037231634	
		h_{err_1}	h_{err_2}	h_{err_3}	

Selanjutnya adalah menghitung nilai *delta* (perubahan) dari *hidden layer*, melalui formula sebagai berikut :

$$\delta h_k = h_err_k * f'(h_k) \dots \dots \dots \quad (22)$$

Dengan formula turunan (*derivative*) dari fungsi aktivasi *sigmoid* sebagaimana tertera pada rumus (2), maka akan dihasilkan perhitungan seperti pada Tabel 3.10.:

Tabel 3.10 Nilai Delta (Perubahan) pada Hidden Layer

	Hidden Layer Error	δh_i
h_err_1	0,173068569	0.02912716
h_err_2	-0,322787224	-0.04814257
h_err_3	-0,037231634	-0.00796421

Selanjutnya adalah menghitung nilai pembaruan untuk setiap bobot pada sinapsis 2, yaitu sinapsis yang menghubungkan *hidden layer* dengan *output layer*, caranya dengan mengalikan neuron pada *hidden layer* dengan *delta output layer*. Perhitungan pembaruan nilai tersebut dapat dianotasikan sebagai berikut :

$$wh_update_{ij} = h_i * \delta y_j \dots \dots \dots \quad (23)$$

Dengan hasil perhitungan nilai pembaruan sinapsis 2 seperti yang terlihat pada Tabel 3.11.

Tabel 3.11 Nilai Pembaruan Sinapsis 2

	δy_1	δy_2
h_1	-0,462435069	0,224399571
h_2	-0,107350729	0,052092627
h_3	-0,40602468	0,19702607

Selanjutnya adalah menghitung nilai pembaruan untuk setiap bobot pada sinapsis 1, yaitu sinapsis yang menghubungkan *input layer* dengan *hidden layer*, caranya dengan mengalikan neuron pada *input layer* dengan *delta hidden layer*. Perhitungan pembaruan nilai tersebut dapat dianotasikan sebagai berikut :

$$w_update_{ij} = x_i * \delta h_j \dots \dots \dots \quad (24)$$

Dengan hasil perhitungan nilai pembaruan sinapsis 1 seperti yang terlihat pada Tabel 3.12.

Tabel 3.12 Nilai Pembaruan Sinapsis 1

	δh_1	δh_2	δh_3
x_1	0.02912716	-0.04814257	-0.00796421
x_2	0.02912716	-0.04814257	-0.00796421
x_3	0	0	0
x_4	0	0	0
x_5	0.02912716	-0.04814257	-0.00796421
x_6	0	0	0

Selanjutnya adalah mengaplikasikan nilai pembaruan tersebut dengan nilai bobot-bobot sebelumnya, dengan cara menjumlahkan masing-masing nilai bobot sebelumnya, dengan nilai pembaruan bobot yang telah dihitung. Perhitungan untuk mendapatkan nilai bobot baru dapat diformulasikan sebagai berikut :

$$w_{ij} = w_{ij} + w_update_{ij} \dots \dots \dots \quad (25)$$

$$wh_{ij} = wh_{ij} + wh_update_{ij} \dots \dots \dots \quad (26)$$

Perhitungan bobot baru untuk sinapsis 1 juga dapat diilustrasikan dengan menjumlahkan nilai bobot pada Tabel 3.3 dengan Tabel 3.12. Nilai bobot baru setelah dilakukan perhitungan pada sinapsis 1 adalah seperti yang tertera pada Tabel 3.13 :

Tabel 3.13 Nilai Bobot Baru pada Input Layer (Sinapsis 1)

	w_1	w_2	w_3
longsor	0,72912716	-0,84814257	0,19203579
korban	0,12912716	-0,54814257	0,29203579
banjir	0,1	0,5	-0,4
tanah	0,7	-0,7	0,4
bencana	0,52912716	-0,24814257	0,29203579
jalan	0,6	-0,2	0,7

Dan nilai bobot baru setelah dilakukan perhitungan pada sinapsis 2 adalah seperti pada Tabel 3.14.

Tabel 3.14 Nilai Bobot Baru pada *Hidden Layer* (Sinapsis 2)

	<i>wh</i> 1	<i>wh</i> 2
h1	-0,36243507	-0,17560043
h2	-0,60735073	0,15209263
h3	-0,90602468	-0,70297393

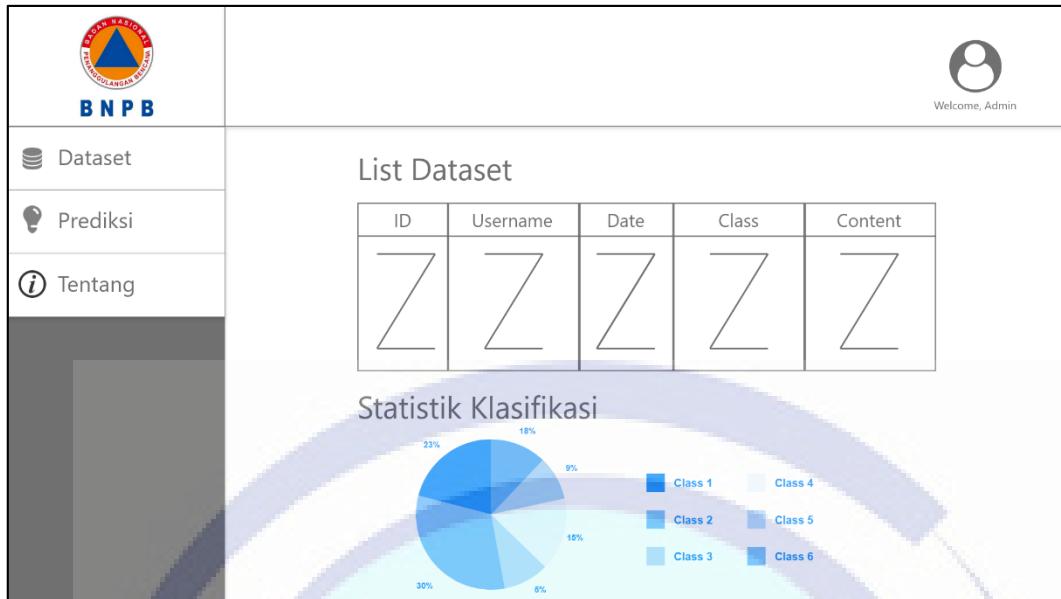
Keseluruhan proses ini dilakukan berulang kali sebanyak *epoch* yang telah ditentukan diawal, dan juga selama nilai *error* rata-rata setiap iterasinya mengalami penurunan, sehingga tercapai nilai bobot yang paling optimal dan paling akurat.

3.6 Rancangan Layar

Rancangan layar merupakan gambaran program berupa *prototype*. Untuk memudahkan penggunaan aplikasi ini oleh pengguna, rancangan layar berikut dibuat dengan mengutamakan *user experience* dan *user interface* yang baik. Berikut adalah rancangan layar *prototype* berdasarkan rancangan program :

3.6.1 *Rancangan Layar List Dataset*

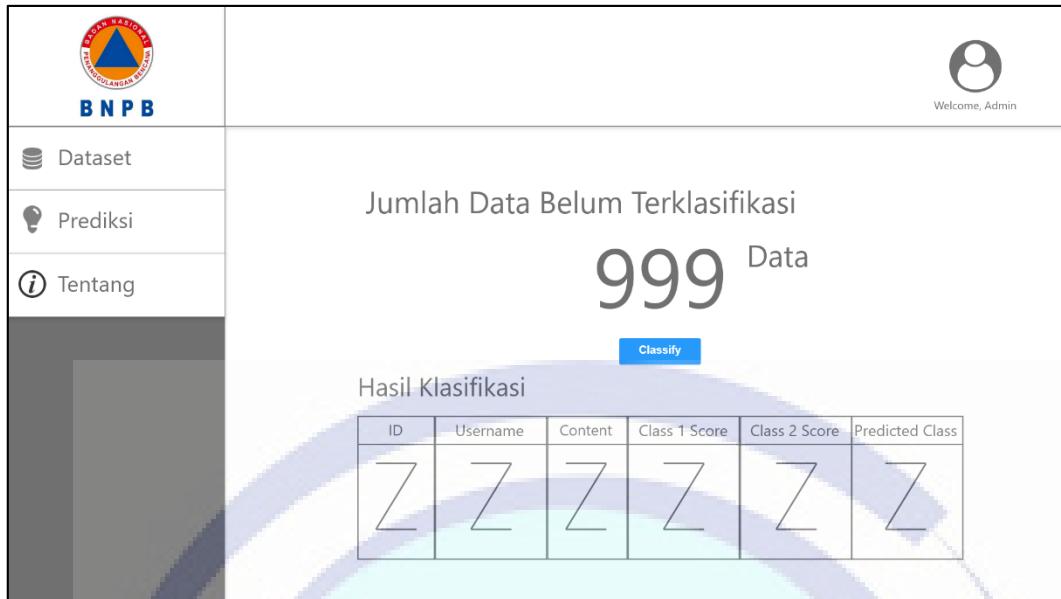
Halaman *list dataset*, yang juga merupakan halaman utama (Home) dari antarmuka aplikasi ini menyajikan seluruh data *tweet* yang tersimpan. Halaman ini juga menyajikan grafik yang menampilkan informasi statistik dari persentase kelas dari seluruh dataset.



Gambar 3.11 Rancangan Layar List Dataset

3.6.2 Rancangan Layar Prediksi

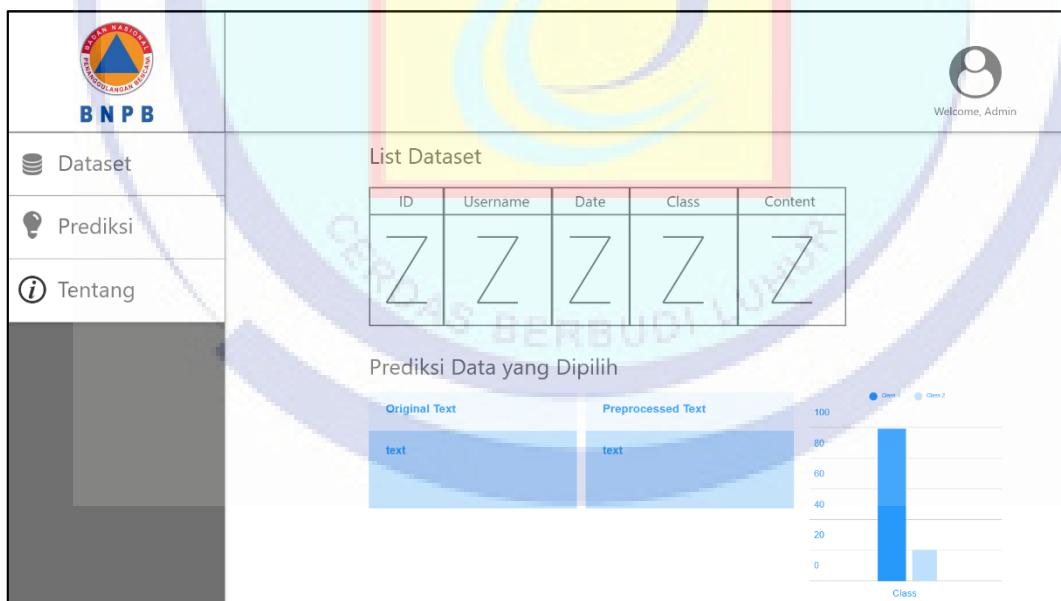
Halaman prediksi data menyajikan kumpulan data yang belum memiliki kelas, atau belum pernah diklasifikasi sebelumnya. Pada halaman ini, pengguna dapat mengklasifikasi secara masif kumpulan data tanpa kelas, melihat hasil perhitungan prediksi serta grafik statistik persentase kelas hasil dari prediksi seluruh dataset tersebut.



Gambar 3.12 Rancangan Layar Halaman Prediksi

3.6.3 Rancangan Layar Halaman Prediksi Spesifik Data

Fitur prediksi spesifik data menyediakan kemampuan untuk memberikan prediksi klasifikasi pada data yang dipilih dari tabel. Fitur prediksi data spesifik ini terdapat di halaman *list dataset*.



Gambar 3.13 Rancangan Layar Halaman Prediksi Spesifik Data

BAB 4

IMPLEMENTASI DAN UJI COBA SOLUSI

4.1 Spesifikasi Sistem

Untuk menunjang performa program, meliputi tahap *training* dan *testing*, diperlukan spesifikasi, sebagai berikut:

4.1.1 *Spesifikasi Perangkat Keras (Hardware)*

Dalam pembuatan aplikasi ini, perangkat keras (*Hardware*) yang digunakan untuk implementasi aplikasi ini sebagai berikut:

1) *Desktop Computer*

Processor	:	Intel Core i3-8100
RAM	:	8Gb
Storage	:	SSD 256GB

4.1.2 *Spesifikasi Perangkat Lunak (Software)*

Dalam pembuatan aplikasi ini, perangkat lunak (*Software*) yang digunakan untuk implementasi aplikasi ini sebagai berikut:

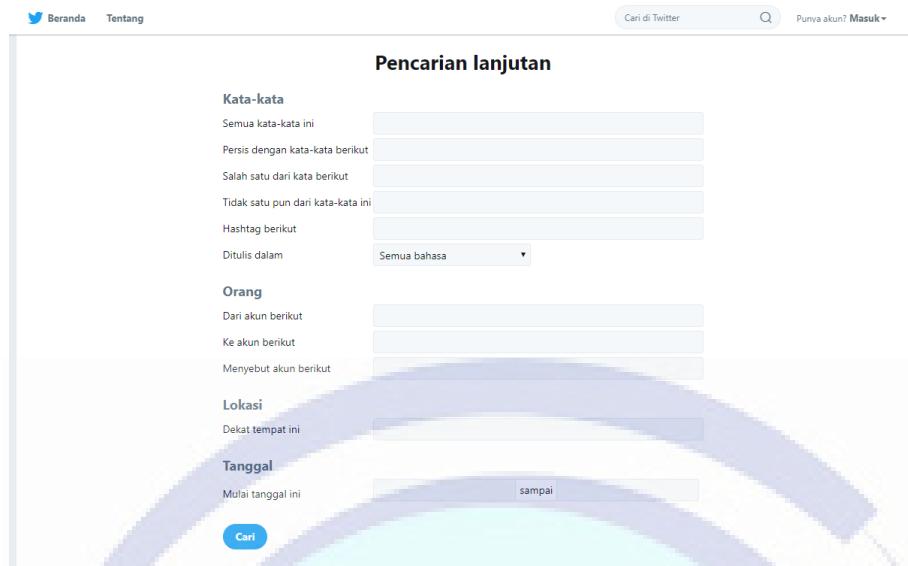
- 1) Python 3.7.3 (32-bit)
- 2) JetBrains PyCharm 2018.3.3
- 3) XAMPP Version: 7.2.13
- 4) MySQL Database
- 5) Web Scraper (Chrome Extension)
- 6) Microsoft Excel 2013
- 7) Microsoft Windows 10

4.2 Pengumpulan Data (*Web Scraping*)

Tahap pengumpulan data merupakan langkah yang pertama kali dilakukan dalam proses klasifikasi data. Pada topik klasifikasi data bencana tanah longsor, sumber data yang digunakan adalah *tweet-tweet* lampau dengan kata kunci “longsor”, yang disediakan oleh Twitter, melalui fitur Advanced Search¹.

Langkah-langkah melaksanakan pengumpulan data dengan *web scraping* , diawali dengan mengakses Twitter Advanced Search .

¹ <https://twitter.com/search-advanced?lang=id>



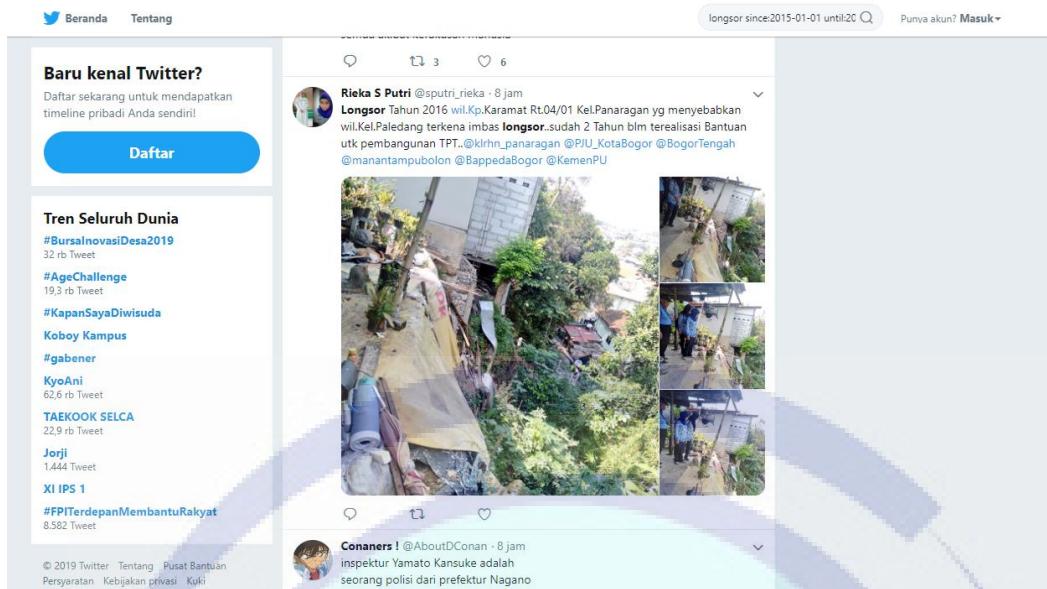
Gambar 4.1 Akses Halaman Twitter Advanced Search

Kemudian isikan kriteria pencarian sesuai dengan kebutuhan topik klasifikasi data, dan masukan rentang waktu *tweet* yang dicari. Pada topik ini, kata kunci yang dicari adalah “longsor” , dengan rentang waktu dari tahun 2015 sampai dengan 2019.



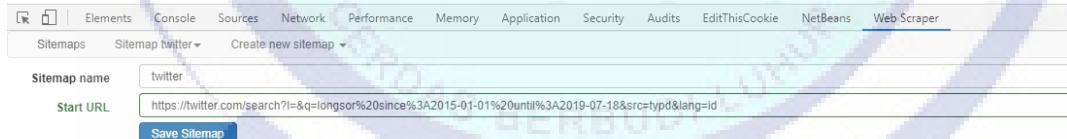
Gambar 4.2 Isi Kriteria Pencarian

Mulai pencarian untuk mendapatkan data yang dicari, hasil yang ditampilkan mengurut berdasarkan waktu *tweet* terbaru.



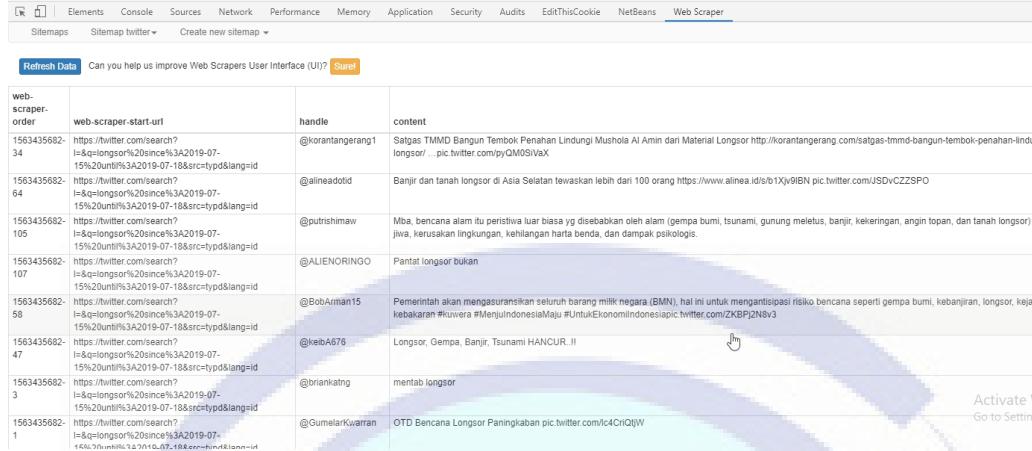
Gambar 4.3 Hasil Pencarian Twitter Advanced Search

Setelah muncul hasil pencarian, salin URL dari Twitter Advanced Search tersebut, dan masukan ke dalam konfigurasi Web Scraper.



Gambar 4.4 Konfigurasi Sitemap Web Scraper

Mulai process *scraping*, dan tunggu hingga halaman Twitter Advanced Search mencapai halaman terbawah, sehingga data *scrape* yang didapat lebih maksimal. Setelah proses *scrape* berhenti, hasil akan didapatkan berupa data yang disajikan dalam tabel.



Gambar 4.5 Hasil *Scraping* Data

Salin data tersebut ke dalam aplikasi Microsoft Excel untuk kemudian disesuaikan dengan format *training data* yang diperlukan dalam proses *training* pada aplikasi. Pada tahap ini, penulis melakukan pelabelan *tweet* secara manual untuk selanjutnya melalui tahap *training*. Selanjutnya simpan file ke dalam format CSV (*Comma-Separated Value*).

A	B	C
username	content	klasifikasi
@vitroglang	Salah satu gambar yg buat miris! Bayangan aja rumah lo di belakangnya ada lobang besar yg sekawtu² bs mengalami longsor dan imbas kena rumah loh #RIPkenyamananmasayarakat #sexykillers . . . PT AdimitraBarataNusantara pic.twitter.com/xkD080fInL	0
@j0bn11	Longsor di Cililin Timbulkan Kerugian di Atas Rp 1 Miliar https://netvib.es/pfAV0K pic.twitter.com/PTJFKzX1	2
@AfifFudS	Hiruk pikuk Pilpres, kita bisa paham melalui acara debat kandidat seperti semalam, namun untuk aksi kemanusiaan, tak perlu berdebat, turun, bantu. Banser Magelang in frame, longsor di Krasak, Jl Magelang-Purworejo. pic.twitter.com/nhy4AH43BhS	0
@Puspen_TNI	Korann 0801/04 Tegalombo Perbaiki Akses jalan Yang Terhambat Akibat Tanah Longsor.....http://tni.mil.id/view-125098-koramili-080104-tegalombo-perbaiki-akses-jalan-yang-terhambat-akibat-tanah-longsor.html#WI1ezXL6A.twitter ...pic.twitter.com/yOMPpEcWMB	2
@tvOneNews	JALUR PENGHUBUNG SUMBAR DAN RIAU TERPUTUS Diketahui curah hujan yang tinggi merupakan penyebab terjadinya longsor di kawasan Kelok Sembilan. Akibatnya, kendaraan yang melintas di kawasan tersebut but harus berhenti & menunggu proses pembersihan material selesai #ONEMINUTETVONEpic.twitter.com/aFJuVZ8HLQ	2
@tni_ad	jajaran Kodam III/SIW melaksanakan kegiatan penyalaman dan evakuasi korban bencana longsor. http://youtu.be/Fp83liBqRHU?a via @YouTube	2
@indosupporter	Hujan deras menyebabkan banjir dan longsor di wilayah Kota Jayapura, Papua. #PrayForJayapura #PrayForPapua pic.twitter.com/HOSbtjeVG	1
@radarmasnetwork	Bangunan SDN Gunungwuled 2 Rawan Longsor https://radarbanyumas.co.id/bangunan-sdn-gunungwuled-2-rawan-longsor/ ...pic.twitter.com/MOANHd84ks	0
@nrizanty1	longsor di freepot, banjir di jalur tan di madiun, tidakkah kalian merasa Allah sedang memberikan peringatan dan tenggang waktu agar yg mendebat ayat Allah segera bertobat. pic.twitter.com/1g1T84C0ia	0
@tempometro	Daerah Rawan Banjir dan Longsor Puncak Bertambah, Ini Sebabnya https://metro.tempo.co/read/1072467/daerah-rawan-banjir-dan-longsor-puncak-bertambah-ini-sebabnya?utm_source=Digital%20Marketing&utm_medium=Twitter&utm_campaign=Metro_Astari ...	2
@INDONESIAinLOVE	Turut berduka cita Dianti Dyah Ayu Cahyani Putri (Putri), meninggal dunia usai berhasil dievakuasi dari longsor bandara Soetta dan dilarikan ke rumah sakit. Putri meninggal dunia dalam usia 25 tahun. pic.twitter.com/oI7lahuJE	2
@detikcom	Saat ini proses evakuasi masih berlangsung di lokasi longsor yang menimbulkan puluhan rumah di Kampung Adat Sinaresmi Sukabumi. https://news.detik.com/jawabarat/4366362/ada-107-jiwा-tinggal-di-area-longsor-kampung-adat-sinaresmi?utm_term=echoboxauto&utm_campaign=detikcomsocmed&utm_medium=oa&utm_content=detikcom&utm_source=Twitter#Echobox=1546271930 ...	2
@KompasTV	[VIDEO] Kendaraan Tertimpa Longsor di Bandara Soetta http://dlvr.it/QF6X2c pic.twitter.com/D2zzL0cLSh	1

Gambar 4.6 Optimasi Format Data Pada Excel dan Pelabelan Manual

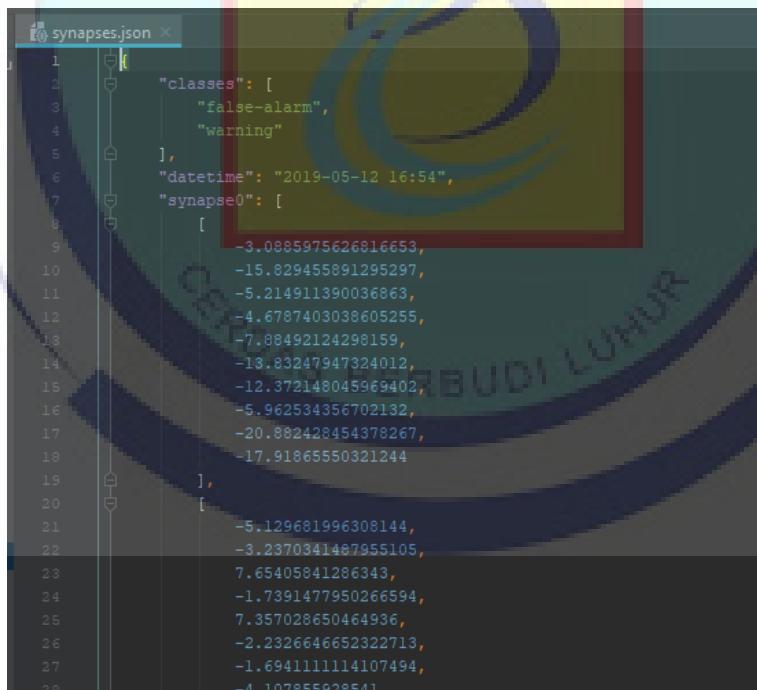
4.3 Tahap Pelatihan Data (*Training Data*)

Setelah data memenuhi format yang diperlukan, jalankan proses *training data* melalui *command-line* sebagai berikut.

```
C:\python_project\neural_network>python train_data.py
Training with 10 neurons, alpha:1, dropout:False
Input matrix: 513x200      Output matrix: 1x2
delta after 10000 iterations:0.03651939841240873
delta after 20000 iterations:0.02919829407009366
delta after 30000 iterations:0.028086689095689444
delta after 40000 iterations:0.026906063391036152
delta after 50000 iterations:0.026775062521666563
delta after 60000 iterations:0.02666866883729467
delta after 70000 iterations:0.026598616973381617
```

Gambar 4.7 Proses *Training Data*

Setelah proses *training data* selesai, aplikasi akan menghasilkan kumpulan bobot yang paling optimal ke dalam sebuah file JSON, seperti gambar berikut.



Gambar 4.8 Hasil Training Data

Kumpulan bobot optimal inilah yang kemudian digunakan untuk mengklasifikasi data-data baru nantinya.

4.4 Tampilan Layar

Berikut ini adalah tampilan layar antarmuka untuk mengklasifikasi data berdasarkan hasil *training* sebelumnya.

4.4.1 Halaman Utama – List Dataset

Halaman utama ini juga sekaligus menampilkan *list* seluruh *dataset* yang tersimpan di dalam *database*.

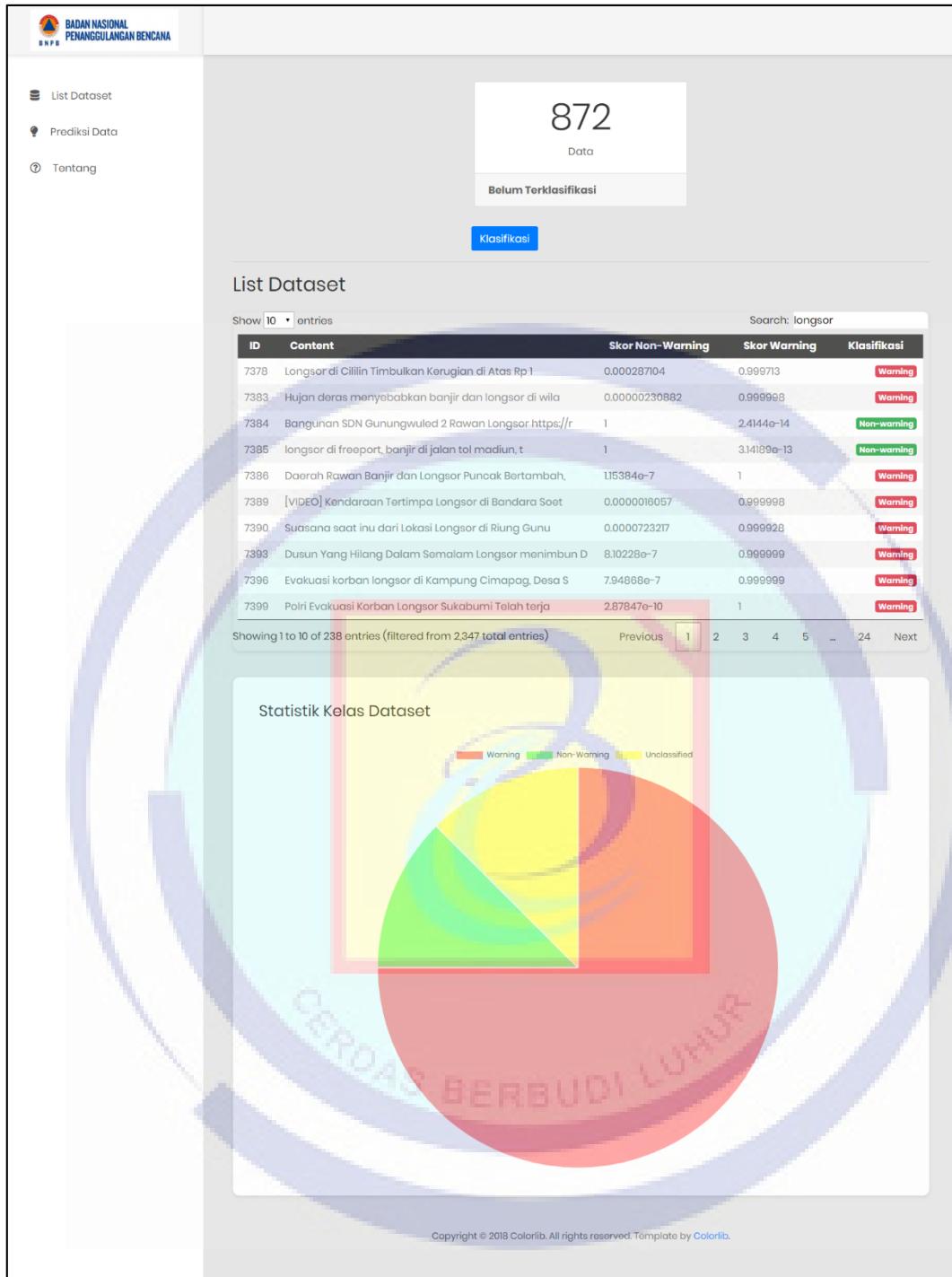




Gambar 4.9 Tampilan Layar Utama

4.4.2 Halaman Prediksi Kelas

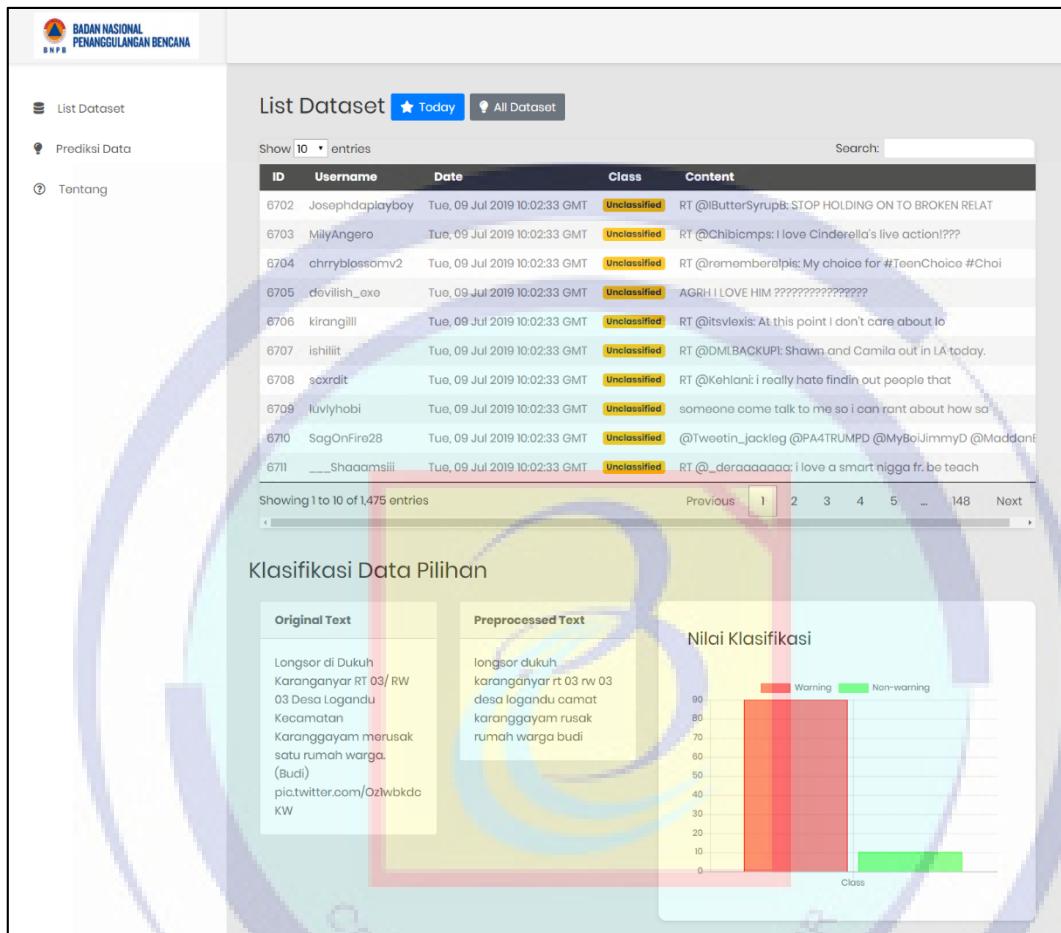
Halaman ini memiliki fungsi untuk mengklasifikasi data-data baru yang didapat dan belum memiliki kelas.



Gambar 4.10 Tampilan Halaman Prediksi

4.4.3 Halaman Prediksi Data yang Dipilih

Halaman ini merupakan bagian dari halaman utama dan halaman prediksi kelas, dan muncul sebagai fitur ketika pengguna memilih data dari tabel, dan kemudian menampilkan klasifikasi datanya.



Gambar 4.11 Halaman Prediksi Data Pilihan

4.5 Pengujian

Tahap pengujian dilakukan dengan mengukur rata-rata dari nilai *error* yang didapat saat *testing*, dan dengan menghitung jumlah prediksi yang benar dan jumlah prediksi yang salah dari keseluruhan data set yang digunakan untuk *testing*, dengan tujuan mendapatkan nilai akurasi.

4.5.1 Menghitung MSE (Mean Square Error)

Perhitungan tingkat kesalahan atau error dilakukan untuk menganalisa jumlah kesalahan yang terjadi antara data aktual dan data peramalan. Perhitungan tingkat kesalahan peramalan menggunakan *mean square error* (MSE) dengan rumus sebagai

berikut:

$$MSE = \frac{1}{n} \sum_{i=1}^n (t_i - y_i)^2$$

Hasil penjumlahan keseluruhan *square error* untuk setiap data *testing* dapat dilihat pada tabel berikut :

Tabel 4.1 Total Square Error Seluruh Data Testing

No.	Konten	Nilai Target		Nilai Prediksi		Square Error
		Non-warning	Warning	Non-warning	Warning	
1	Salah satu gambar yg buat miris! B:	1	0	0.999798893	0.000201099	4.04E-08
2	Longsor di Cililin Timbulkan Kerug	0	1	0.000287104	0.999712895	8.24E-08
3	Hiruk pikuk Pilpres, kita bisa paha	1	0	0.999523032	0.000477104	2.28E-07
4	Koramil 0801/04 Tegalombo Perb:	0	1	0.000308192	0.999692151	9.49E-08
5	JALUR PENGHUBUNG SUMB/	0	1	8.24E-07	0.999999176	6.78E-13
6	jajaran Kodam III/Slw melaksanak	0	1	7.91E-07	0.999999209	6.26E-13
7	Hujan deras menyebabkan banjir d	0	1	2.30882E-06	0.999997691	5.33E-12
8	Bangunan SDN Gunungwuled 2 Re	1	0	1	2.41E-14	1.46E-28
9	longsor di freeport, banjir di jalan tc	1	0	1	3.14E-13	9.87E-26
10	Daerah Rawan Banjir dan Longsor	0	1	1.15E-07	0.999999885	1.33E-14
11	Turut berduka cita Dianti Dyah Ayi	0	1	8.36E-09	0.999999992	6.99E-17
12	Saat ini proses evakuasi masih berk	0	1	7.90E-07	0.99999921	6.25E-13
13	[VIDEO] Kendaraan Tertimpa Lori	0	1	1.6057E-06	0.999998393	2.58E-12
14	Suasana saat inu dari Lokasi Longs	0	1	7.23217E-05	0.99992767	5.23E-09
15	Edy Rahmayadi Pimpin Langsung F	0	1	8.06E-07	0.999999194	6.49E-13
16	selamat siang sahabat bara kiki tahu	1	0	0.999912897	8.70501E-05	7.58E-09
17	Dusun Yang Hilang Dalam Semalar	0	1	8.10E-07	0.99999919	6.56E-13
18	Berkat SWADAYA dan GOTONI	0	1	7.50E-07	0.99999925	5.63E-13
19	Situasi Proses evakuasi pembersihan	0	1	7.91E-07	0.999999209	6.26E-13
20	Evakuasi korban longsor di Kampu	0	1	7.95E-07	0.999999205	6.32E-13
...
241	BMKG terus memantau aktivitas G	1	0	0.999999837	1.63E-07	2.66E-14
242	Hendak Pergi Mengaji, Kakak Ber	1	0	0.999524227	0.000475852	2.26E-07
243	Koordinasi & Sinergitas dgn Pemk:	0	1	0.000561486	0.999438012	3.16E-07
244	1. Info @KemenPU : Kepolisian R	1	0	0.999988027	1.19695E-05	1.43E-10
245	Mobil yang Tertimbun Longsor Soe	0	1	7.88E-07	0.999999212	6.21E-13
246	Menanggapi peristiwa tanah longso	1	0	0.999695175	0.000304628	9.29E-08
247	Mohon ijin pak..longsor sekarang d	0	1	6.85E-08	0.999999936	4.40E-15
248	Hujan deras yang mengguyur kawa	0	1	6.68E-07	0.999999332	4.47E-13
249	ibarat ramalan cuaca: dibeberapa te	1	0	0.999999385	6.15E-07	3.78E-13
250	Hujan deras mengguyur Jakarta, ke	0	1	9.28E-07	0.999999072	8.61E-13
Total Square Error :						8.78E+00

Keterangan :

- * 1.0 E+00 = 1
- * 1.0 E+01 = 10
- * 1.0 E-01 = 0.1

Dengan demikian, hasil perhitungan MSE adalah sebagai berikut :

$$MSE = \frac{8,78}{500} = 0,01756$$

Berdasarkan hasil perhitungan, bahwa tingkat kesalahan dari algoritme *backpropagation* pada kasus ini sangat kecil, yaitu hanya sebesar 0,01756.

4.5.2 Menghitung Akurasi Algoritme Backpropagation

Perhitungan tingkat akurasi dari algoritme *backporpagation* yang telah melalui tahap *training*, adalah dengan membandingkan label target dengan label yang dihasilkan dari hasil prediksi, dan menghitung berapa banyak jumlah data yang label prediksinya sesuai dengan label target dari keseluruhan data *testing*. Hasil perhitungan akurasi dapat dilihat pada tabel berikut :



Tabel 4.2 Perhitungan Tingkat Akurasi Algoritme Backpropagation

No.	Konten	Label		Status
		Target	Prediksi	
1	Salah satu gambar yg buat miris! Ba...	Non-Warning	Non-Warning	Cocok
2	Longsor di Cililin Timbulkan Kerugi...	Warning	Warning	Cocok
3	Hiruk pikuk Pilpres, kita bisa pahai ...	Non-Warning	Non-Warning	Cocok
4	Koramil 0801/04 Tegalombo Perbaiki...	Warning	Warning	Cocok
5	JALUR PENGHUBUNG SUMBAWA DAN SUL...	Warning	Warning	Cocok
6	jajaran Kodam III/Slw melaksanakan ...	Warning	Warning	Cocok
7	Hujan deras menyebabkan banjir di da...	Warning	Warning	Cocok
8	Bangunan SDN Gunungwuled 2 Raya ter...	Non-Warning	Non-Warning	Cocok
9	longsor di freeport, banjir di jalan tc...	Non-Warning	Non-Warning	Cocok
10	Daerah Rawan Banjir dan Longsor di ...	Warning	Warning	Cocok
11	Turut berduka cita Dianti Dyah Ayu ...	Warning	Warning	Cocok
12	Saat ini proses evakuasi masih berlangsung ...	Warning	Warning	Cocok
13	[VIDEO] Kendaraan Tertimpa Longsor di ...	Warning	Warning	Cocok
14	Suasana saat ini dari Lokasi Longsor di ...	Warning	Warning	Cocok
15	Edy Rahmayadi Pimpin Langsung Penanganan ...	Warning	Warning	Cocok
16	selamat siang sahabat bara kiki taher ...	Non-Warning	Non-Warning	Cocok
17	Dusun Yang Hilang Dalam Semalar di ...	Warning	Warning	Cocok
18	Berkat SWADAYA dan GOTONG YAYA ...	Warning	Warning	Cocok
19	Situasi Proses evakuasi pembersihan ...	Warning	Warning	Cocok
20	Evakuasi korban longsor di Kampung ...	Warning	Warning	Cocok
...
240	Salju longsor ditambah turunnya salju ...	Non-Warning	Non-Warning	Cocok
241	BMKG terus memantau aktivitas Gempa ...	Non-Warning	Non-Warning	Cocok
242	Hendak Pergi Mengaji, Kakak Beri ...	Non-Warning	Non-Warning	Cocok
243	Koordinasi & Sinergitas dgn Pemkab ...	Warning	Warning	Cocok
244	1. Info @KemenPU : Kepolisian RI ...	Non-Warning	Non-Warning	Cocok
245	Mobil yang Tertimbun Longsor Soekarno ...	Warning	Warning	Cocok
246	Menanggapi peristiwa tanah longsor di ...	Non-Warning	Non-Warning	Cocok
247	Mohon ijin pak..longsor sekarang di ...	Warning	Warning	Cocok
248	Hujan deras yang mengguyur kawasan ...	Warning	Warning	Cocok
249	Ibarat ramalan cuaca: dibeberapa hari ...	Non-Warning	Non-Warning	Cocok
250	Hujan deras mengguyur Jakarta, ke ...	Warning	Warning	Cocok
Total Akurasi :		245 / 250 = 98 %		

Berdasarkan hasil perhitungan tersebut, dapat disimpulkan bahwa akurasi dari penerapan algoritme *backpropagation* dalam kasus ini sangat tinggi, yaitu mencapai 98 %.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan perumusan masalah penelitian yang telah dipaparkan sebelumnya, dapat ditarik kesimpulan sebagai berikut :

1. Arsitektur *artificial neural network* yang digunakan terdiri dari *input layer* dengan 200 *neuron*, satu *hidden layer* dengan 10 *neuron*, dan *output layer* dengan 2 *neuron* (2 kelas), *non-warning* dan *warning*.
2. Nilai rata-rata kesalahan (*mean square error*) dari 500 data yang dilakukan uji coba (*testing*), menunjukkan nilai yang kecil yaitu sebesar 0,01756.
3. Nilai uji akurasi pada 500 data *testing*, menunjukkan hasil akurasi yang sangat tinggi yaitu mencapai 98,2 %, atau 491 dari 500 data cocok.

5.2 Saran

Dari hasil penelitian dan pembahasan yang telah dilakukan, maka saran yang dapat dikemukakan yaitu sebagai berikut :

1. Pada implementasi selanjutnya dengan topik yang sama, dapat digunakan beberapa kelas lagi agar klasifikasi menjadi lebih luas dan mencakup seluruh jenis bencana.
2. Tingkat akurasi dapat lebih ditingkatkan dengan menggabungkan algoritme *artificial neural network* lainnya, seperti *recurrent neural network*, atau *convolutional neural network*.

DAFTAR PUSTAKA

- Ahmad, A., Putri, P. M., Alifah, W., Gunawan, I. dan . S. (2019) “Analisis Jaringan Syaraf Tiruan Metode Backpropagation Dalam Memprediksi Ketersediaan Komoditas Beras Berdasarkan Provinsi Di Indonesia,” *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, 2(1), hal. 48–60. doi: 10.31598/jurnalresistor.v2i1.348.
- Anang Sugeng, C. (2016) “Pengaruh media sosial terhadap perubahan sosial masyarakat di Indonesia,” *Jurnal ilmu sosial & ilmu politik diterbitkan oleh Fakultas Ilmu Sosial & Politik, Universitas Tulungagung*, 9(1), hal. 140–157. Tersedia pada: <http://www.jurnal-unita.org/index.php/publiciana/article/download/79/73>.
- Haykin, S. dan Widrow, B. (2003) “*Least Mean Square Adaptive Filters.*” Wiley Online Library.
- Jin, P., Yue, Z., Chen, X. dan Xia, Y. (2016) “Bag-of-Embeddings for Text Classification,” *IJCAI International Joint Conference on Artificial Intelligence*, 2016-Janua, hal. 2824–2830.
- Johnston, J., Dumbill, E. dan St. Laurent, S. (2001) “*Programming Web Services with XML-RPC.*” O’Reilly Media, Inc. Tersedia pada: <http://books.google.com/books?hl=en&lr=&id=ALo1LxID5q0C&oi=fnd&pg=PR9&dq=Programming+Web+Services+with+SOAP&ots=zP9NNJpmvd&sig=afUPO0pkc-HwWfNP1S3Wsl3Fcrk>.
- Kartika, A., Irawan, B. dan Triyanto, D. (2016) “Prediksi Wilayah Rawan Kebakaran Hutan dengan Metode Jaringan Syaraf Tiruan Propagasi Balik (Study Kasus : Daerah Kabupaten Kuburaya),” *Jurnal Coding, Sistem Komputer Untan*, 4(2), hal. 66–75.
- Mandowara, J. dan Jain, A. (2016) “Text Classification by Combining Text Classifiers to Improve the Efficiency of Classification,” *International Journal of Computer Application*, 6(2), hal. 126–129. Tersedia pada: <http://rspublication.com/ijca/2016/april16/15.pdf>.
- Moghaddam, A. H., Moghaddam, M. H. dan Esfandyari, M. (2016) “Predicción del

- índice del mercado bursátil utilizando una red neuronal artificial,” *Journal of Economics, Finance and Administrative Science*. AEDV, 21(41), hal. 89–93. doi: 10.1016/j.jefas.2016.07.002.
- Nazief, B., Adriani, M., Asian, J., Williams, H. E. dan Tahaghoghi, S. M. M. (2005) “Stemming Indonesian,” *Conferences in Research and Practice in Information Technology Series*, 38(4), hal. 307–314. doi: 10.1145/1316457.1316459.
- Ostad Ali Askari, K., Shayannejad, M. dan Ghorbanizadeh Kharazi, H. (2017) “Artificial Neural Network for Modeling Nitrate Pollution of Groundwater in Marginal Area of Zayandeh-rood River, Isfahan, Iran,” *KSCE Journal of Civil Engineering*, 21(1), hal. 134–140. doi: 10.1007/s12205-016-0572-8.
- Putri, W. S. R., Nurwati, N. dan Budiarti S., M. (2016) “Pengaruh Media Sosial Terhadap Perilaku Remaja,” *Prosiding Penelitian dan Pengabdian kepada Masyarakat*, 3(1). doi: 10.24198/jppm.v3i1.13625.
- Rao, J. dan Su, X. (2005) “A Survey of Automated Web Service Composition Methods,” hal. 43–54. doi: 10.1007/978-3-540-30581-1_5.
- Riswanto, R. L., Sutikno dan Indriyati (2014) “Aplikasi Prediksi Jumlah Penderita Penyakit Demam Berdarah Denga di Kota Semarang Menggunakan Jaringan Syaraf Tiruan Backpropagation,” 5, hal. 19–27.
- Sanner, M. F. (1999) “Python: a programming language for software integration and development,” *Journal of molecular graphics & modelling*, 17(1), hal. 57–61. Tersedia pada: <http://www.ncbi.nlm.nih.gov/pubmed/10660911>.
- Tala, F. Z. (2017) “A Study of Stemming Effect on Information Retrieval in Bahasa Indonesia,” *Jurnal Teknosains*, 6(2), hal. 113. doi: 10.22146/teknosains.26972.
- Zareapoor, M. dan K. R, S. (2015) “Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection,” *International Journal of Information Engineering and Electronic Business*, 7(2), hal. 60–65. doi: 10.5815/ijieeb.2015.02.08.

LAMPIRAN

SURAT KETERANGAN SELESAI RISET



LISTING PROGRAM

1. UBLTextMining.py

```
2. import re
import csv
from nltk.tokenize import TweetTokenizer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import numpy as np
import datetime
import json

class NeuralNetwork:
    corpus = [] #input layer
    kbki = []
    classes = ['nonwarning', 'warning']
    training_sentences = []
    stop_words = []

    X = []
    Y = []

    def __init__(self, data_source =
'lib\\data_training.csv',
corpus_source='lib\\input_layer.txt',stopwords_source='lib\\
stopwords.txt', input_classes = []):
        if input_classes :
            self.classes = input_classes
        self.load_training_data(data_source)
        self.load_corpus(corpus_source)
        self.load_kbki()
        self.load_stop_words(stopwords_source)
    # ----- Main Function -----
    def start_training_process(self):
        self.preprocess_sentence()
        self.train()

    def start_mse_process(self):
        testing_data = []
        i=1
        training_sentences_temp = self.training_sentences
        for doc in training_sentences_temp:
            tweet_data = {}
            classes_in_binary = [0] * len(self.classes)
            classes_in_binary[int(doc[1])] = 1
            classes_to_compare = np.array(classes_in_binary)

            prediction = self.predict(sentence=doc[0])
            prediction_list = prediction.tolist()
            layer_2_error = np.array(classes_to_compare -
prediction)
            last_mean_error = np.mean(np.abs(layer_2_error))
            mse = np.square(last_mean_error)

            tweet_data['content'] = doc[0]
            tweet_data['target_0']=classes_in_binary[0]
            tweet_data['target_1']=classes_in_binary[1]
```

```
        tweet_data['prediction_0']=prediction_list[0]
        tweet_data['prediction_1']=prediction_list[1]
        tweet_data['mse'] = mse
        testing_data.append(tweet_data)
        progress = i/len(training_sentences_temp) * 100
        i+=1
        print('Progress : ', progress, ' %')

    return testing_data

def start_hitung_akurasi(self):
    testing_data = []
    i = 1

    training_sentences_temp =
    self.training_sentences[:500]
    for doc in training_sentences_temp:
        tweet_data = {}

        prediction = self.predict(sentence=doc[0])
        label_prediksi = 1 if prediction[1] >
        prediction[0] else 0

        tweet_data['content'] = doc[0]
        tweet_data['target'] = 'Warning' if int(doc[1])
        == 1 else 'Non-Warning'
        tweet_data['prediksi'] = 'Warning' if
        label_prediksi == 1 else 'Non-Warning'
        if label_prediksi == int(doc[1]):
            tweet_data['status'] = 'Cocok'
            tweet_data['true'] = 1
        else:
            tweet_data['status'] = 'Tidak Cocok'
            tweet_data['true'] = 0

        tweet_data['total_data'] =
        len(training_sentences_temp)

        testing_data.append(tweet_data)
        progress = i / len(training_sentences_temp) *
        100
        i += 1
        print('Progress : ', progress, ' %')
    return testing_data

# ----- Single Function -----
def load_training_data(self, data_source =
'lib\\data_training.csv'):
    with open(data_source) as csvfile:
        readCSV = csv.reader(csvfile, delimiter=';')
        for row in readCSV:

    self.training_sentences.append((row[1],row[2]))

def load_corpus(self, data_source = 'lib\\corpus.txt'):
    with open(data_source) as f:
```

```
        self.corpus = f.read().split('\n')

    def load_kbbi(self, data_source = 'lib\\corpus.txt'):
        with open(data_source) as f:
            self.kbbi = f.read().split('\n')

    def load_stop_words(self, data_source =
'lib\\stopwords.txt'):
        with open(data_source, 'r') as f:
            self.stop_words = f.read().split('\n')

    def bag_of_binary_words(self, sentence,
show_details=False):
        factory = StemmerFactory()
        stemmer = factory.create_stemmer()

        # CASE FOLDING
        kalimat = sentence.lower()

        # TOKENIZATION
        kalimat = re.sub('[^A-Za-z0-9# ]+', '', kalimat)
        kalimat = kalimat.split()

        # STOPWORD REMOVAL
        for word in kalimat:
            if word in self.stop_words:
                kalimat.remove(word)

        # STEMMING SASTRAWI
        stemmed = []
        for word in kalimat:
            stemmed.append(stemmer.stem(word))
        #return np.array(stemmed)

        words_in_binary = []
        for word in self.corpus:
            if word in stemmed:
                words_in_binary.append(1)
                if(show_details):
                    print ("Found keyword: {}".format(word))
            else:
                words_in_binary.append(0)
        return np.array(words_in_binary)

    def preprocess_sentence_only(self, sentence):
        factory = StemmerFactory()
        stemmer = factory.create_stemmer()

        # CASE FOLDING
        kalimat = sentence.lower()

        # TOKENIZATION
        kalimat = re.sub('[^A-Za-z0-9# ]+', '', kalimat)
        kalimat = kalimat.split()

        # STOPWORD REMOVAL
```

```
for word in kalimat:
    if word in self.stop_words:
        kalimat.remove(word)

# STEMMING SASTRAWI
stemmed = []
for word in kalimat:
    stemmed.append(stemmer.stem(word))

for word in stemmed:
    if word not in self.kbki:
        stemmed.remove(word)

return np.array(stemmed)

def preprocess_sentence(self):
    for doc in self.training_sentences:

self.x.append(self.bag_of_binary_words(sentence=doc[0]))

classes_in_binary = [0] * len(self.classes)
classes_in_binary[int(doc[1])] = 1

self.y.append(classes_in_binary)

# compute sigmoid nonlinearity
def sigmoid(self, x):
    output = 1/(1+np.exp(-x))
    return output

# convert output of sigmoid function to its derivative
def sigmoid_output_to_derivative(self, output):
    return output*(1-output)

def predict(self, sentence, show_details=False):
    with open("lib\\synapses.json") as data_file:
        synapse = json.load(data_file)
        synapse_0 = np.asarray(synapse['synapse0'])
        synapse_1 = np.asarray(synapse['synapse1'])
    x = self.bag_of_binary_words(sentence=sentence,
show_details=show_details)
    if show_details:
        print ("sentence:", sentence, "\n bow:", x)
    # input layer is our bag of words
    l0 = x
    # matrix multiplication of input and hidden layer
    l1 = self.sigmoid(np.dot(l0, synapse_0))
    # print("Layer Hidden 1 : ")
    # print(l1)
    # output layer
    l2 = self.sigmoid(np.dot(l1, synapse_1))
    return l2

def train(self,hidden_neurons=10, alpha=1,
epochs=5000000, dropout=False, dropout_percent=0.5):
    print ("Training with %s neurons, alpha:%s,
```

```

dropout:%s %s" % (hidden_neurons, str(alpha), dropout,
dropout_percent if dropout else ''))
    print ("Input matrix: %sx%s      Output matrix: %sx%s"
% (len(self.X), len(self.corpus), 1, len(self.classes)))

    last_mean_error = 1
    # randomly initialize our weights with mean 0
    synapse_0 = 2 * np.random.random((len(self.corpus),
hidden_neurons)) - 1
    synapse_1 = 2 * np.random.random((hidden_neurons,
len(self.classes))) - 1

    prev_synapse_0_weight_update =
np.zeros_like(synapse_0)
    prev_synapse_1_weight_update =
np.zeros_like(synapse_1)

    synapse_0_direction_count = np.zeros_like(synapse_0)
    synapse_1_direction_count = np.zeros_like(synapse_1)

    i = 0
    for j in iter(range(epochs + 1)):

        # Feed forward through layers 0, 1, and 2
        layer_0 = np.array(self.X)
        layer_1 = self.sigmoid(np.dot(layer_0,
synapse_0))

        if (dropout):
            layer_1 *=
np.random.binomial([np.ones((len(self.X), hidden_neurons))],
1 - dropout_percent)[0] * (
                1.0 / (1 - dropout_percent))

        layer_2 = self.sigmoid(np.dot(layer_1,
synapse_1))

        # how much did we miss the target value?
        layer_2_error = self.y - layer_2

        if (j % 10000) == 0 and j > 5000:
            # if this 10k iteration's error is greater
            # than the last iteration, break out
            if np.mean(np.abs(layer_2_error)) <
last_mean_error:
                print ("delta after " + str(j) + " "
iterations:" + str(np.mean(np.abs(layer_2_error))))
                last_mean_error =
np.mean(np.abs(layer_2_error))
            else:
                print ("break:",
np.mean(np.abs(layer_2_error)), ">", last_mean_error)
                break

        # in what direction is the target value?
        # were we really sure? if so, don't change too

```

```

much.

        layer_2_delta = layer_2_error *
self.sigmoid_output_to_derivative(layer_2)

        # how much did each l1 value contribute to the
l2 error (according to the weights)?
        layer_1_error = layer_2_delta.dot(synapse_1.T)

        # in what direction is the target l1?
        # were we really sure? if so, don't change too
much.
        layer_1_delta = layer_1_error *
self.sigmoid_output_to_derivative(layer_1)

        synapse_1_weight_update =
(layer_1.T.dot(layer_2_delta))
        synapse_0_weight_update =
(layer_0.T.dot(layer_1_delta))
        #print('update:',synapse_1_weight_update,
synapse_0_weight_update)
        print(synapse_1_weight_update)

        if (j > 0):
            synapse_0_direction_count += np.abs(
                ((synapse_0_weight_update > 0) + 0) -
                ((prev_synapse_0_weight_update > 0) + 0))
            synapse_1_direction_count += np.abs(
                ((synapse_1_weight_update > 0) + 0) -
                ((prev_synapse_1_weight_update > 0) + 0))

            synapse_1 += alpha * synapse_1_weight_update
            synapse_0 += alpha * synapse_0_weight_update

            prev_synapse_0_weight_update =
synapse_0_weight_update
            prev_synapse_1_weight_update =
synapse_1_weight_update

            # print("Progress : {}".format(i/epochs*100))
            # i+=1

        now = datetime.datetime.now()

        # persist synapses
        synapse = {'synapse0': synapse_0.tolist(),
'synapse1': synapse_1.tolist(),
'datetime': now.strftime("%Y-%m-%d
%H:%M"),
'classes': self.classes
}
        synapse_file = "lib\\synapses.json"

        with open(synapse_file, 'w') as outfile:
            json.dump(synapse, outfile, indent=4,
sort_keys=True)
            print ("saved synapses to:", synapse_file)

```

```
def test_json(self):
    # persist synapses
    now = datetime.datetime.now()
    synapse = {
        'datetime': now.strftime("%Y-%m-%d %H:%M"),
        'classes': self.classes
    }
    synapse_file = "lib\\synapses.json"

    with open(synapse_file, 'w') as outfile:
        json.dump(synapse, outfile, indent=4,
sort_keys=True)
        print ("saved synapses to:", synapse_file)

def classify(self, sentence, ERROR_THRESHOLD = 0,
show_details=False):
    results = self.predict(sentence, show_details)

    results = [[i, r] for i, r in enumerate(results) if
r > ERROR_THRESHOLD]
    #print(results)
    results.sort(key=lambda x: x[1],
reverse=True) # fungsi lambda
tersebut, maksudnya jadikan index 1 sebagai patokan sortnya
    # return_results = [[self.classes[r[0]], r[1]] for r
in results]
    return_results = [{"class_id":r[0], "class": self.classes[r[0]], "score": r[1]} for r
in results]
    #print ("%s \n classification: %s" % (sentence,
return_results))
    return return_results
```

3. Extract_katadasar.py

```
from tools.UBLTextMining import NeuralNetwork
import csv

nn = NeuralNetwork()
with open('lib\\katadasar.txt') as f:
    corpus_indo = f.read().split()

training_data = []
with open('lib\\longsor_raw_data.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=';')
    for row in readCSV:
        training_data.append(row[2])

len_training = len(training_data)
i = 1
corpus_dari_training_data = []
for kalimat in training_data :
    check_kalimat = nn.preprocess_sentence_only(sentence=kalimat)
```

```
for kata in check_kalimat :
    if kata not in corpus_dari_training_data:
        if kata in corpus_indo:
            corpus_dari_training_data.append(kata)
print('Loading {}%'.format(i/len_training*100))
i= i+1

with open('lib\\corpus_new.txt', 'w') as f:
    for item in corpus_dari_training_data:
        f.write("%s\n" % item)

print('Done')
```

4. Define_input_layer.py

```
5. from tools.UBLTextMining import NeuralNetwork
import csv
import json

nn = NeuralNetwork()
katadasar_dict = {}
with open('lib\\corpus_new.txt') as f:
    corpus_indo = f.read().split()

for word in corpus_indo:
    katadasar_dict[word] = 0

training_data = []
with open('lib\\longsor_raw_data.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=';')
    for row in readCSV:
        training_data.append(row[2])

len_training = len(training_data)
i = 1

for kalimat in training_data:
    kalimat_tokenize =
nn.preprocess_sentence_only(sentence=kalimat)
    for kata in kalimat_tokenize:
        if kata in katadasar_dict:
            x = katadasar_dict[kata]
            katadasar_dict[kata] = x + 1
    print('Loading {}%'.format(i / len_training * 100))
    i = i + 1

katadasar_score_file = "source\\katadasar_score.json"
with open(katadasar_score_file, 'w') as outfile:
    json.dump(katadasar_dict, outfile, indent=4,
sort_keys=True)
print ("saved score to:", katadasar_score_file)

sorted_katadasar = sorted(katadasar_dict,
key=katadasar_dict.get, reverse=True)
```

```
i = 0
with open('lib\\input_layer.txt', 'w') as f:
    for item in sorted_katadasar:
        if i <= 200:
            f.write("%s\n" % item)
            i = i +1
        else:
            break

print ("Input Layer Listed in.")

# Loop tiap record buat dapetin jumlah tiap word increment
# Sort dict
# Write ke JSON
```

6. Run.py

```
from web_service import app
app.run(debug=True)
```

7. Stream_data.py

```
8. from __future__ import absolute_import, print_function
from tools.UBLTextMining import NeuralNetwork
from tweepy import OAuthHandler, Stream, StreamListener
import json
import mysql.connector
from datetime import datetime
# Go to http://apps.twitter.com and create an app.
# The consumer key and secret will be generated for you
after
consumer_key="Wj62IMs0E3Mk82CqscYBYXKWq"
consumer_secret="v8xWUlFyAwwyrNxnVQP0m9TUJtzZGKr5FL6pMl1D4aW
GIZREdI"

# After the step above, you will be redirected to your app's
page.
# Create an access token under the "Your access token"
section
access_token="588890544-
NxMMx97OmFl1zvh8GW2Q4ITfzlenk0dhdCruFaG8"
access_token_secret="GSvloCJrywYDu4EwJJzD9nq3Mr0OTfmmcHhWh8W
cdcR9"

class StdOutListener(StreamListener):
    """ A listener handles tweets that are received from the
stream.
    This is a basic listener that just prints received
tweets to stdout.
    """
    db = mysql.connector.connect(
        host="localhost",
        user="root",
```

```
        passwd="",
        database="dataset_skripsi"
    )

    cursor = db.cursor()
    def on_data(self, data):
        #print(data)
        # i=1
        # f= open("lib\\mantap.json", "w+")
        # f.write(data)
        # print ("Saved {} tweet".format(i))
        # i+= 1
        tweet_data = json.loads(data)
        var_screen_name = tweet_data['user']['screen_name']
if 'user' in tweet_data else ''
        var_name = tweet_data['user']['name'] if 'user' in
tweet_data else ''
        var_text = tweet_data['text'] if 'text' in
tweet_data != None else ''
        ts = tweet_data['timestamp_ms'] if 'timestamp_ms' in
tweet_data else 0
        var_timestamp = datetime.fromtimestamp(int(ts) / 1e3)

        nn = NeuralNetwork()
        classify_result = nn.classify(sentence=var_text)
        classified = {res["class"] : res['score'].item() for
res in classify_result}
        classified['predicted'] = '0' if classified["non-
warning"] > classified["warning"] else '1'

        self.cursor.execute('''INSERT INTO training_data
(username, name, text, created_at, class, score_0, score_1)
VALUES(%s, %s, %s, %s, %s, %s )''' , (
            var_screen_name,
            var_name,
            var_text,
            var_timestamp,
            classified['predicted'],
            str(classified['non-warning']),
            str(classified['warning'])
        ))
        self.db.commit()
        print (var_text)
        return True

    def on_error(self, status):
        print(status)

if __name__ == '__main__':
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    stream = Stream(auth, l)
    stream.filter(track=['cinta'])
```

9. Train_data.py

```
from tools.UBLTextMining import NeuralNetwork

nn = NeuralNetwork(data_source = 'lib\\data_training.csv')
nn.start_training_process()
```

10. Web_service/__init__.py

```
11. from flask import Flask
     from flaskext.mysql import MySQL
     from flask_cors import CORS

     app = Flask(__name__)
     CORS(app)
     db = MySQL()
     app.config['MYSQL_DATABASE_USER'] = 'root'
     app.config['MYSQL_DATABASE_PASSWORD'] = ''
     app.config['MYSQL_DATABASE_DB'] = 'dataset_skripsi'
     app.config['MYSQL_DATABASE_HOST'] = 'localhost'
     db.init_app(app)

     from web_service.training_data.views import datatraining
     from web_service.classify.views import classifyprocedure
     app.register_blueprint(datatraining)
     app.register_blueprint(classifyprocedure)
```

12. Web_service/training_data/models.py

```
from web_service import db
from tools.UBLTextMining import NeuralNetwork
class DataTraining():

    def all(self, page=1):
        conn = db.connect()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM training_data ORDER BY
created_at DESC")
        result = cursor.fetchall()
        data_train = [{cursor.description[i][0]:item[i] for i in
range(len(cursor.description))} for item in result]

        return data_train

    def today(self):
        conn = db.connect()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM training_data WHERE
DATE_FORMAT(created_at, '%Y-%m-%d') = DATE_FORMAT(sysdate(), '%Y-
%m-%d') ORDER BY created_at DESC")
```

```
        result = cursor.fetchall()
        data_train = [({cursor.description[i][0]: item[i] for i in
range(len(cursor.description))}) for item in result]
        cursor.close()
        return data_train

    def find(self, id):
        conn = db.connect()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM training_data where id=%s
ORDER BY created_at DESC"%(id))
        result = cursor.fetchall()
        data_train = [({cursor.description[i][0]: item[i] for i in
range(len(cursor.description))}) for item in result]
        cursor.close()
        return data_train

    def search(self, keyword):
        conn = db.connect()
        cursor = conn.cursor()
        search_keyword = "%" + keyword + "%"
        cursor.execute("SELECT * FROM training_data WHERE text
LIKE %s OR name like %s OR username like %s OR created_at like %s
ORDER BY created_at DESC", (search_keyword, search_keyword,
search_keyword, search_keyword))
        result = cursor.fetchall()
        data_train = [({cursor.description[i][0]: item[i] for i in
range(len(cursor.description))}) for item in result]
        cursor.close()
        return data_train

    def insert(self, username, name, text):
        conn = db.connect()
        cursor = conn.cursor()
        status = cursor.execute("INSERT INTO training_data
(username, name, text) VALUES(%s, %s, %s)", (username, name, text))
        if status:
            conn.commit()
            return True
        else:
            return False

    def update(self, id, username, name, text, kelas):
        conn = db.connect()
        cursor = conn.cursor()
        status = cursor.execute("UPDATE training_data SET
username=%s, name=%s, text=%s, class=%s WHERE id=%s", (username,
name, text, kelas, id))
        if status:
            conn.commit()
            cursor.close()
            return True
        else:
            return False

    def delete(self, id):
```

```
conn = db.connect()
cursor = conn.cursor()
status = cursor.execute("DELETE FROM training_data WHERE
id=%s", (id))
if status:
    conn.commit()
    cursor.close()
    return True
else:
    return False

def statistikAll(self):
    conn = db.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT class as class_id, count(*) as
total FROM training_data GROUP BY class")
    result = cursor.fetchall()
    cursor.close()
    data_train = [{cursor.description[i][0]: item[i] for i in
range(len(cursor.description))} for item in result]
    statistik = {}
    for item in data_train:
        if item["class_id"] == '0':
            label = "nonwarning"
        elif item["class_id"] == '1':
            label = "warning"
        else:
            label = "unclassified"
        statistik[label] = item["total"]
    # for item in data_train:
    #     if item['class_id'] is '':
    #         class_name='Unclassified'
    #     else:
    #         #
    class_name=NeuralNetwork.classes[int(item['class_id'])]
    #     data_train['class_name'] = class_name
    return statistik

def statistikSearch(self, keyword):
    conn = db.connect()
    cursor = conn.cursor()
    search_keyword = "%" + keyword + "%"
    cursor.execute("SELECT class as class_id, count(*) as
total FROM training_data WHERE text LIKE %s OR name like %s OR
username like %s OR created_at like %s GROUP BY class",
                  (search_keyword, search_keyword,
                   search_keyword, search_keyword))
    result = cursor.fetchall()
    cursor.close()
    data_train = [{cursor.description[i][0]: item[i] for i in
range(len(cursor.description))} for item in result]
    statistik = {}
    for item in data_train:
        if item["class_id"] == '0':
            label = "nonwarning"
        elif item["class_id"] == '1':
```

```
        label = "warning"
    else:
        label = "unclassified"
    statistik[label] = item["total"]
# for item in data_train:
#     if item['class_id'] is '':
#         class_name='Unclassified'
#     else:
#         #
class_name=NeuralNetwork.classes[int(item['class_id'])]
#         data_train['class_name'] = class_name
return statistik

def total(self):
    conn = db.connect()
    cursor = conn.cursor()
    cursor.execute(
        "SELECT COUNT(0) as total FROM training_data"
    )
    result = cursor.fetchall()
    data_train = [{cursor.description[i][0]: item[i] for i in range(len(cursor.description))} for item in result]
    cursor.close()
    return data_train

def total_bySearch(self, keyword):
    conn = db.connect()
    cursor = conn.cursor()
    search_keyword = "%" + keyword + "%"
    cursor.execute(
        "SELECT COUNT(0) as total FROM training_data WHERE
text LIKE %s OR name like %s OR username like %s OR created_at
like %s ORDER BY created_at DESC", (search_keyword, search_keyword,
search_keyword, search_keyword))
    result = cursor.fetchall()
    data_train = [{cursor.description[i][0]: item[i] for i in range(len(cursor.description))} for item in result]
    cursor.close()
    return data_train
```

13. Web_service/training_data/views.py

```
14. import json
from flask import request, jsonify, Blueprint, abort
from flask.views import MethodView
from web_service import app
from web_service.training_data.models import DataTraining

datatraining = Blueprint('datatraining', __name__)

@datatraining.route('/')
@datatraining.route('/home')
def home():
    return "Welcome to the Data Home."
```

```
@datatraining.route('/data/today')
def today():
    data_training = DataTraining.today(app)
    return jsonify(data_training)

@datatraining.route('/data/statistik/')
def statistikAll():
    statistik = DataTraining.statistikAll(app)
    return jsonify(statistik)

@datatraining.route('/data/statistik/<keyword>')
def statistikSearch(keyword=None):
    statistik =
    DataTraining.statistikSearch(app, keyword=keyword)
    return jsonify(statistik)

@datatraining.route("/data/total/", methods=["GET"])
def getTotal():
    result = DataTraining.total(app)
    return jsonify(result[0])

@datatraining.route("/data/total/<keyword>",
methods=["GET"])
def getTotalBySearch(keyword):
    result = DataTraining.total_bySearch(app,
keyword=keyword)
    return jsonify(result[0])

class DataTrainingView(MethodView):

    def get(self, id=None, keyword=None, page=1):
        if not id and not keyword:
            data_training = DataTraining.all(app)
        elif not id:
            data_training = DataTraining.search(app,
keyword)
        else:
            data_training = DataTraining.find(app, id)

        return jsonify({"data":data_training})
    # return jsonify(data_training)

    def post(self):
        username = request.form.get('username')
        name = request.form.get('name')
        text = request.form.get('text')

        if DataTraining.insert(app, username, name, text):
            return jsonify(
                {"status" : "Ok"})
        )
    else:
        return jsonify(
            {"status": "Failed"})
        )
```

```
def put(self, id):
    username = request.form.get('username')
    name = request.form.get('name')
    text = request.form.get('text')
    kelas= request.form.get('class')

    if DataTraining.update(app, id=id,
        username=username, name=name, text=text, kelas=kelas):
        return jsonify(
            {"status": "Ok"})
    )
else:
    return jsonify(
        {"status": "Failed"})
)

def delete(self, id):
    if DataTraining.delete(app, id=id):
        return jsonify(
            {"status": "Ok"})
    )
else:
    return jsonify(
        {"status": "Failed"})
)

datatraining_view =
DataTrainingView.as_view('datatraining_view')
app.add_url_rule(
    '/data/', view_func=datatraining_view, methods=['GET',
    'POST'])
)
app.add_url_rule(
    '/data/<int:id>', view_func=datatraining_view,
    methods=['GET', 'PUT', 'DELETE'])
)
app.add_url_rule(
    '/data/search/<keyword>', view_func=datatraining_view,
    methods=['GET'])
)
```

15. Web_service/classify/models.py

```
from web service import db
from tools.UBLTextMining import NeuralNetwork

class ClassifyProcedure:

    def classifyText(self, text):
        nn = NeuralNetwork()
        preprocessed_text = ""
        for item in nn.preprocess_sentence_only(text):
```

```

        preprocessed_text += (item + " ")
    classify_result = nn.classify(sentence=text)
    classified = {res["class"]: res['score'].item() for res in
classify_result}
    classified['predicted'] = '0' if classified["non-warning"] > classified["warning"] else '1'
    return {"original":text,"preprocessed":preprocessed_text,"class":classified}

    def classifyDataset(self, id):
        conn = db.connect()
        cursor = conn.cursor()
        cursor.execute(
            "SELECT * FROM training_data WHERE id = %s", (id)
        )
        result = cursor.fetchall()
        tweet = [({cursor.description[i][0]: item[i] for i in range(len(cursor.description))}) for item in result]

        nn = NeuralNetwork()
        preprocessed_text = ""
        for item in nn.preprocess_sentence_only(tweet[0]["text"]):
            preprocessed_text += (item + " ")
        classify_result = nn.classify(sentence=tweet[0]["text"])
        classified = {res["class"]: res['score'].item() for res in
classify_result}
        classified['predicted'] = '0' if classified["non-warning"] > classified["warning"] else '1'
        cursor.execute(
            '''UPDATE training_data set class=%s,score_0=%s,
score_1=%s WHERE id=%s''', (
                classified['predicted'],
                str(classified['non-warning']),
                str(classified['warning']),
                id
            )
        )
        conn.commit()
        cursor.close()
        return {"original": tweet[0]["text"], "preprocessed": preprocessed_text, "class": classified}

    def update_score(self):
        conn = db.connect()
        cursor = conn.cursor()
        nn = NeuralNetwork()
        cursor.execute("SELECT * FROM training_data ")
        # cursor.execute("SELECT * FROM training_data WHERE text like '%longson%'")
        result = cursor.fetchall()
        cursor.close()
        data_train = [({cursor.description[i][0]: item[i] for i in range(len(cursor.description))}) for item in result]

        debug_classified=[]
        for data in data_train:

```

```
        classify_result = nn.classify(sentence=data['text'])
        # classified=
{res['class']: {"class_id": res['class_id'],
"score": res['score'].item()} for res in classify_result}
        classified= {res["class"] : res['score'].item() for res
in classify_result}
        classified['predicted'] = '0' if classified["non-
warning"] > classified["warning"] else '1'
        # debug_classified.append(classified)
        # score_0 = classified['non-warning']['score']
        # score_1 = classified['warning']['score']
        #
        # predicted_class = classified['non-
warning']['class_id'] if score_0 > score_1 else
classified['warning']['class_id']
        conn = db.connect()
        cursor = conn.cursor()
        cursor.execute(
            '''UPDATE training_data set class=%s,score_0=%s,
score_1=%s WHERE id=%s''', (
                classified['predicted'],
                str(classified['non-warning']),
                str(classified['warning']),
                data['id']
            ))
        conn.commit()
        cursor.close()
    return {"status": "Berhasil"}
    # return {"data": debug_classified}
```

16. Web_service/classify/views.py

```
17. import json
from flask import request, jsonify, Blueprint, abort
from flask.views import MethodView
from web_service import app
from web_service.classify.models import ClassifyProcedure

classifyprocedure = Blueprint('classifyprocedure', __name__)

@classifyprocedure.route("/update_score_all",
methods=["GET"])
def update_score_all():
    result = ClassifyProcedure.update_score(app)
    return jsonify(result)

@classifyprocedure.route("/classify/<id>", methods=["GET"])
def classifyDataset(id):
    result = ClassifyProcedure.classifyDataset(app, id=id)
    return jsonify(result)

class ClassifyView(MethodView):

    def post(self):
```

```
text = request.form.get('text')

result = ClassifyProcedure.classifyText(app, text)
return jsonify({"data":result})

classify_view = ClassifyView.as_view('classify_view')
app.add_url_rule(
    '/classify', view_func=classify_view, methods=['POST']
)
```

