PREDIKSI *RATING* OTOMATIS BERDASARKAN *REVIEW* RESTORAN PADA APLIKASI *ZOMATO* DENGAN MENGGUNAKAN *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Diajeng Tania Ananda P
NIM: 151550207111169

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PREDIKSI RATING OTOMATIS BERDASARKAN *REVIEW* RESTORAN PADA APLIKASI ZOMATO DENGAN MENGGUNAKAN *EXTREME LEARNING MACHINE* (ELM)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Diajeng Tania Ananda P
NIM: 155150207111169

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Mei 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Imam Cholissodin, S.Si, M.Kom	Candra Dewi, S.Kom, M.Sc
NIK: 201201 850719 1 001	NIP: 19771114 200312 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Maret 2019

Diajeng Tania Ananda P

NIM: 155150207111169

PRAKATA

Pertama-tama kami panjatkan puji dan syukur atas kehadirat Allah SWT yang telah memberikan rahmat, taufik, serta hidayahnya sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Prediksi *Rating* Otomatis Berdasarkan *Review* Restoran Pada Aplikasi Zomato Dengan Menggunakan *Extreme Learning Machine* (ELM)" ini dengan baik.

Dengan selesainya skripsi ini, tidak sedikit bantuan yang diterima dari banyak pihak. Oleh karena itu, penulis ingin menyampaikan rosa hormat dan terima kasih kepada:

1. Allah SWT atas limpahan berkah dan hidayah serta bimbingan dalam pengerjain skripsi.
2. Bapak Imam Cholissodin, S.Si, M.Kom dan Ibu Candra Dewi, S.Kom, M.Sc selaku dosen pembimbing skripsi yang telah memberikan arahan terhadap penulis dengan penuh kesabaran sehingga skripsi ini dapat terselesaikan.
3. Bapak Agus Wahyu Widodo, S.T, M.Sc selalu Ketua Program Studi Teknik Informatika.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
5. Kedua orang tua tercinta Bapak Dwi Ananda dan Ibu Wahyu Yulianti serta keluarga besar yang telah mendoakan, memberikan kasih sayang serta memberikan dukungan berupa moril maupun materil, hingga terselesaikan skripsi ini.
6. Aqli Akbar yang selalu memotivasi, memberikan doa, semangat, dan dukungan kepada penulis hingga terselesaikannya skripsi ini
7. Teman-teman dekat, Adhelia Regita, Arsya Monica, Annisa Rizkyani, Putri Bunga, Putri Stephanie, Desy Andriani, Afina Putri, Carlista Naba, Krishna Yudha, Rangga, Rama dan Rafi yang telah memberikan bantuan, semangat serta doa hingga terselesaikannya skripsi ini.
8. Teman-teman penulis, Ghea Faradiba, Anjani Dinda, Natasya Kriswandhany, Nadia Ingrida, Dea Sahirah, Adinda Saptadirdja, Astrid Meidiana, dan Fiona Ayesha selaku teman penulis yang berada di Jakarta yang selalu memberikan semangat dan dukungan kepada penulis, sehingga penulis dapat menyelesaikan skripsi ini.
9. Audia Refanda dan Robih Dini yang telah menjadi teman berdiskusi selama pengerjaan skripsi agar terselesaikannya skripsi ini
10. Teman-teman Teknik Informatika angkatan 2015, seluruh dosen, dan civitas akademik Fakultas Ilmu Komputer yang telah banyak memberi

bantuan dan dukungan selama penulis menempuh studi dan selama penyelesaian skripsi ini.

Demi kesempurnaan skripsi ini masih banyak kekurangan, sehingga memerlukan kritik dan saran yang membangun bagi penulis. Akhir kata penulis berharap skripsi ini dapat memberikan manfaat bagi semua pihak yang menggunakannya

Malang, 1 Maret 2019

Penulis

taniaananda09@yahoo.com

ABSTRAK

Diajeng Tania, Prediksi *Rating* Otomatis berdasarkan *Review* Restoran pada Aplikasi Zomato dengan menggunakan *Extreme Learning Machine* (ELM)

Pembimbing: Imam Cholissodin, S.Si, M.Kom dan Candra Dewi, S.Kom, M.Sc

Seiring berkembangnya zaman, perkembangan teknologi semakin berkembang. Salah satunya aplikasi untuk mencari informasi mengenai restoran di Jakarta yaitu Zomato. Zomato merupakan aplikasi yang menyediakan informasi dari berbagai restoran, fasilitas, harga, *review*, dan *rating* dari restoran tersebut. Masyarakat dapat memberikan *review* dan *rating* pada restoran tersebut. Data *review* tersebut berguna untuk pengguna sebelum ke restoran tersebut. Data *review* tersebut terkadang belum disertai *rating* sehingga membuat pemilik restoran mendapatkan masalah dalam mengklasifikasikan *review* kedalam *rating* untuk melakukan evaluasi kedepannya pada restoran tersebut. Pada penelitian ini membantu untuk mengklasifikasikan *review* kedalam *rating*. Pengujian metode ini menggunakan prediksi dengan metode *Extreme Learning Machine* (ELM). Proses prediksi ini menggunakan tahapan *pre-processing*, pembobotan kata dengan TF-IDF, dan perhitungan metode *Extreme Learning Machine* (ELM). Terdapat tahapan-tahapan pada metode ELM antara lain normalisasi, proses *training*, dan proses *testing*. Metode ELM ini menghasilkan akurasi sebesar 80,01% Dengan jumlah k yaitu 10 menggunakan *hidden neuron* sebanyak 25 dengan Interval bobot -0,5 hingga 0,5 fungsi aktivasi Sigmoid biner. Dapat disimpulkan bahwa metode ELM dapat menyelesaikan masalah prediksi dengan cukup baik.

Kata kunci: prediksi *rating*, review, *Extreme Learning Machine*

ABSTRACT

Diajeng Tania Ananda, Automatic Rating Predictions based on restaurant review in Zomato Application by using Extreme Learning Machine (ELM).

Supervisors: Imam Cholissodin, S.Si, M.Kom and Candra Dewi, S.Kom, M.Sc

In this modern culture, technology advancement are growing better than we ever discovered before. One of the apps we use to search for information about restaurant in Jakarta are known as Zomato. Zomato is an application that provides various information about a restaurant from it facility, price, review, and rating. Users of The Zomato App can input various information that people haven't aware of about the restaurant into the app. Besides of inputting information into the app, Users of The Zomato App can also input a review and rating of a specific restaurant. The data review is used as an information about the restaurant for the potential customer from The Zomato App but sometimes the data review doesn't yet include a restaurant rating. This lack of misinformation will surely make the restaurant owner to occure some difficulties such as improving the restaurant services status for future outcomes. This research helps to classifying the review into the rating. Test protocol of this research are using a prediction with Extreme Learning Machine (ELM) Methods as it core. The prediction process however are build from a several steps such as pre–processing, word weighting with TF-IDF, and Extreme Learning Machine (ELM) Method calculations. Test result of The ELM parameter provides accuracy result 80,01% with k=10 amount hidden neuron 25 Interval weights -0.5 until 0,5 using function activation Sigmoid biner. We have come to conclusion were ELM method could positively solve the prediction problem exquisitely.

Keywords: Rating prediction, Review, Extreme Learning Machine

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Zaman sekarang teknologi semakin canggih, semakin berkembangnya zaman. Pada era sekarang informasi mengenai berbagai hal sangat mudah untuk didapatkan. Salah satunya untuk mencari Restoran di kota-kota besar seperti Jakarta, Surabaya, Semarang, Bandung, dan lain sebagainya. Pencarian restoran biasanya dilakukan untuk memudahkan konsumen melihat menu yang ditawarkan, harga, dan fasilitas yang disediakan. Sebelum memilih restoran biasanya konsumen dapat melihat *review* dari konsumen lain yang memberikan pengalamannya di sosial media.

Zomato merupakan salah satu pengembangan aplikasi *mobile* yang semakin canggih untuk membantu masyarakat mencari restoran berdasarkan kategori-kategori tertentu. Zomato dapat digunakan melalui *mobile* dan dapat juga diakses melalui *website* Zomato. Aplikasi tersebut memudahkan masyarakat khususnya di kota-kota besar untuk mencari restoran, *coffeeshop*, *live music*, dan lain sebagainya dari kota yang dipilih. Konsumen dapat memilih lokasi yang diinginkan, selain itu dapat melihat menu dari restoran tersebut, dapat memberikan *review*, dan masyarakat dapat melihat *rating* dari *review* pengalaman konsumen lain. Dalam setiap *review* pada aplikasi Zomato tersebut sudah disertai dengan *rating*. Menurut (Jong, 2011) *rating* merupakan representasi tingkat kepuasan dari konsumen. Adanya *rating* yang diberikan oleh konsumen untuk memberikan gambaran dari setiap restoran. Dengan banyaknya data yang melimpah dari berbagai konsumen dan beberapa *review* juga belum adanya *rating* maka membuat pihak restoran mengalami kesulitan dalam mengklasifikasikan *review* kedalam *rating* dan mengetahui kekurangan dari restoran tersebut. Oleh karena itu, perlu adanya sistem untuk membantu mempercepat prediksi *rating* berdasarkan *review* konsumen agar restoran dapat dilakukan evaluasi untuk meningkatkan kualitas tersebut.

Salah satu metode yang digunakan untuk memprediksi sesuatu telah banyak dilakukan dengan berbagai penerapan metode mulai dari metode prediksi yang sederhana hingga yang kompleks seperti menggunakan *Naïve Bayes, Backpropagation*, *Extreme Learning Machine, Support Vector Machine*. Salah satunya adalah *Extreme Learning Machine*, ELM dilakukan untuk menanggulangi kesalahan-kesalahan dari jaringan syaraf tiruan *feedforward* yang terpenting dalam hal *learning speed*. Dalam segi kualitas dan waktu, maka metode ELM lebih unggul dibandingkan metode *Support Vector Machine* dan *Backpropagation* (Huang, et al., 2006).

Penelitian sebelumnya yang membahas mengenai prediksi untuk mengurangi resiko kerugian investasi, dikarenakan nilai tukar valas selalu berubah-ubah setiap waktu. Data yang digunakan untuk penelitian tersebut adalah data kurs jual dan kurs beli antara mata uang Dollar Amerika (USD) dengan mata uang Rupiah (IDR).

Pada penelitian ini menggunakan 5 *input layer* dan 25 *hidden layer* diperoleh nilai *error* sebesar 0,0000368 pada kurs jual dan 0,001596 pada kurs beli.

Penelitian sebelumnya pada prediksi pemotongan *laser* pada *Heat Affected Zone* (HAZ) menggunakan *Extreme Learning Machine* (ELM) dibandingkan dengan *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GP). Dari perbandingan dari ketiga algoritme tersebut, ELM memiliki nilai *error* terendah yaitu 0,0046 dibandingkan ANN dan GP.

Berdasarkan uraian dari penjelasan sebelumnya, membuktikan bahwa metode *Extreme Learning Machine* mampu menghasilkan nilai *error* yang kecil dari suatu permasalahan. Oleh sebab itu maka peneliti ingin mengusulkan sebuah penelitian yang berjudul Prediksi *Rating* Otomatis pada *Review* Restoran dengan metode *Extreme Learning Machine (ELM)*, diharapkan dengan menggunakan algoritme ELM penelitian mendapatkan akurasi yang tinggi dan dapat membantu menemukan kekurangan restoran untuk meningkatkan fasilitas dari restoran tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang mengenai permasalahan prediksi *rating* otomatis berdasarkan adanya beberapa *review* restoran, maka dapat didapatkan rumusan masalah yaitu:

1. Bagaimana kemampuan dari metode *Extreme Learning Machine* (ELM) untuk prediksi *rating* otomatis pada *review* restoran?
2. Bagaimana nilai evaluasi yang dihasilkan metode *Extreme Learning Machine* (ELM) untuk prediksi *rating* otomatis pada *review* restoran?

1.3 Tujuan

Dari penjabaran rumusan masalah diatas, maka tujuan yang hendak diperoleh dalam penelitian ini yaitu diantaranya:

1. Mengetahui kemampuan dari metode *Extreme Learning Machine* untuk memprediksi *rating* otomatis berdasarkan *review* restoran.
2. Mengetahui nilai evaluasi yang dihasilkan metode *Extreme Learning Machine* (ELM) untuk prediksi *rating* otomatis pada *review* restoran.

1.4 Manfaat

Manfaat dari pengujian metode untuk prediksi *rating* maka didapatkan manfaat sebagai berikut:

1. Membantu masyarakat untuk menentukan pilihan restoran dengan melihat *rating* yang dihasilkan.
2. Diharapkan dapat membantu pihak restoran dalam mengetahui respon masyarakat sehingga dapat memudahkan upaya untuk mengevaluasi kekurangan yang ada.

1.5 Batasan Masalah

Berikut batasan masalah dalam prediksi *rating* otomatis berdasarkan *review* yang digunakan untuk inti dari permasalahan yang ada, sebagai berikut:

1. Data yang digunakan yaitu data *offline* dari aplikasi Zomato, data tersebut berupa *review* restoran beserta *rating*.
2. Data yang digunakan berjumlah 150 data
3. Klasifikasi yang dilakukan yaitu *review* berbahasa Indonesia.
4. Terdapat 5 kelas dalam penelitian ini yaitu *rating* 1 hingga *rating* 5.

1.6 Sistematika Pembahasan

Berikut sub bab sistematika pembahasan yang digunakan untuk memberikan gambaran umum pada penelitian yang akan dibahas, sebagai berikut:

BAB I. PENDAHULUAN

Pada bab pendahuluan terdapat latar belakang terjadinya permasalahan yang muncul, rumusan masalah yang ingin diselesaikan, tujuan dari penelitian ini, manfaat penelitian, batasan masalah yang digunakan untuk mengetahui batasan-batasan dari penelitian ini, dan sistematika pembahasan yang digunakan untuk mengetahui gambaran umum pada penelitian ini.

BAB II. LANDASAN KEPUSTAKAAN

Bab yang membahas mengenai penelitan-penelitian sebelumnya dan pembahasan *review* mengenai teori-teori yang berkaitan dengan permasalahan dalam penelitian ini serta teori-teori pendukung.

BAB III. METODOLOGI PENELITIAN

Pada bab ini terdiri dari tahapan yang akan dibangun untuk penelitian mencakup tipe penelitian yang dilakukan, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, 3eknik analisis hasil, dan implementasi algoritme

BAB IV. PERANCANGAN

Bab ini berisi analisis dan perancangan membahas tentang analisa kebutuhan dari prediksi *rating* berdasarkan ulasan pada aplikasi Zomato.

BAB V. IMPLEMENTASI

Pada bab ini berisi tentang implementasi dan pembahasan mengenai prediksi *rating* otomatis berdasarkan *review* restoran pada aplikasi Zomato.

BAB VI. PENGUJIAN DAN ANALISIS

Bab pengujian dan analisis ini terkait dengan pengujian yang akan dilakukan pada sistem prediksi *rating* berdasarkan otomatis berdasarkan

review restoran pada aplikasi Zomato yang telah diimplementasikan serta analisis dari pengujian tersebut.

BAB VII. PENUTUP

Bab Penutup merupakan bab yang menganalisis kesimpulan yang di dapatkan dalam skripsi serta saran untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini membas mengenai tentang penelitian yang dilakukan sebelumnya yang bertujuan untuk menganalisa sebagai rujukan pustaka. Pada penelitian sebelumnya juga membahas mengenai metode yang digunakan dan objek penelitian. Tiap pustaka dianalisa oleh objek pada penelitian tersebut, metode yang dilakukan, dan hasil penelitian yang didapatkan.

2.1 Kajian Pustaka

Salah satu metode dalam melakukan prediksi adalah metode *Extreme Learning Machine* (ELM). Algoritme untuk *feedforward* berlapis tunggal jaringan syaraf untuk masalah *non linear*. Menurut (Huang, et al., 2006) ELM dapat disebut algoritme yang sederhana dibandingkan dengan algoritme jaringan syaraf tiruan lainnya.

Hasil penelitian pertama yang menggunakan data penjualan peritel *fashion*. Dalam penelitian mereka untuk memprediksi penjualan busana untuk jumlah penjualan akhir. Dalam penelitian ini menggunakan metode ELM dan BPNN dengan hasil menggunakan metode *Extreme Learning Machine (ELM)* peramalan praktis penjualan *fashion* lebih stabil daripada algoritme BPNN dalam faktor (ukuran, warna, dll). Penelitian ini berhasil menunjukan bahwa ELM lebih baik ketika menggunakan peramalan praktis penjualan *fashion*.

Selanjutnya pustaka kedua mengenai prediksi yang telah dilakukan sebelumnya oleh (Afifah & Med, 2015). Pada penelitian ini dilakukan untuk memprediksi prospek investasi saham di masa depan, para investor prediksi ini sangat penting untuk mengukur kinerja seluruh saham menggunakan metode *Extreme Learning Machine menghasilkan Mean Square Error* sebesar 0,0082 dengan 7 hidden layer.

Pada pustaka ketiga mengenai prediksi nilai tukar mata uang USD dengan Rupiah dengan data sekunder merupakan kurs jual dan kurs beli antara mata uang Dollar (USD) dengan Rupiah (IDR) untuk memprediksi nilai tukar mata uang kedepannya. Penelitian ini menggunakan metode *Extreme Learning Machine* dengan nilai *error* 0,000368 pada kurs jual dan 0,001596 pada kurs beli. Nilai *error* terendah ini menunjukan bahwa model telah optimal dan dapat digunakan untuk memprediksi.

Pustaka selanjutnya melakukan prediksi pemotongan laser menggunakan *Heat Affected Zone* (HAZ), data yang digunakan berupa kekuatan laser, kecepatan memotong, tekanan gas. Pada hasil prediksi tersebut menggunakan ELM, *Artificial Neural Network* (ANN), dan *Genetic Algorithm* (GP). Hasil prediksi tersebut menghasilkan nilai *error* ELM sebesar 0,0046, ANN sebesar 0,0169, dan GP sebesar 0,0342. Perbandingan nilai *error* terendah pada ELM.

Penelitian kelima merupakan analisis sentimen berdasarkan ulasan pengguna Samsung di Twitter menggunakan metode *Extreme Learning Machine* dan *Ensemble Feature.* Hasil penelitian tersebut menghasilkan akurasi sebesar 42,857% dengan fungsi aktivasi Sigmoid Bipolar dengan 5000 hidden neuron.

Untuk perbandingan objek penelitian serta metode yang digunakan terdapat pada Tabel 2.1

Tabel 2.1 Kajian Pustaka

No	Pustaka	Objek	Metode	Keluaran
		Parameter	**Proses**	**Hasil Penelitian**
1	(Sun, Z.-L, et al., 2008)	- Parameter input : warna dan ukuran pakaian	- *Extreme Learning Machine* (ELM) - *Backpropagation Neural Network* (BPNN)	- Prediksi praktis penjualan fashion menggunakan *Extreme Learning Machine* lebih stabil daripada algoritme BPNN dalam faktor (ukuran, warna)
2	(Afifah & Med, 2015)	- Parameter input : data indeks harga saham gabungan periode harian	- *Extreme Learning Machine*	- Hasil prediksi harga saham di masa depan yang menghasilkan nilai *Mean Square Error* sebesar 0,0033 dengan 7 *hidden layer*.
3	(Septiana, 2017)	- Parameter input berupa data kurs beli dan kurs jual dari 4	- *Extreme Learning Machine* (ELM)	- Prediksi nilai tukar mata uang dengan hasil *error* 0,000368 pada kurs

No	Pustaka	Objek	Metode	Keluaran
		Parameter	**Proses**	**Hasil Penelitian**
		Januari 2016 hingga 31 Januari 2017		jual dan 0,001596 pada kurs beli
4	(Anicic, et al., 2017)	- Parameter input berupa kekuatan laser, kecepatan memotong, tekanan gas	- *Extreme Learning Machine* - *Artificial Neural Network* - *Genetic Algorithm*	- Hasil dari prdiksi tersebut nilai error yang dihasilkan ELM paling kecil sebesar 0,0046 dibandingkan dengan ANN dan GP
5.	(Rausanfita, 2017)	- Paramer input berupa review pelanggan Samsung pada twitter	- *Extreme Learning Machine* - *Ensemble feature*	- Hasil analisis sentiment Twitter Samsung menggunakan *hidden neuron* sebanyak 5000 tingkat akurasi 42,857%

2.2 *Rating*

Rating merupakan bagian terpenting yang akan dilihat oleh konsumen yaitu mencerminkan dengan benar baik atau buruknya dalam peringkat *rating*.Salah satunya pada penelitian ini menggunakan *review* beserta *rating* restoran. *Rating* dari *review* restoran dapat membantu konsumen dan pemilik restoran dalam mengetahui kualitas restoran, baik itu menu ataupun pelayanannya, layak atau tidak untuk dipilih. *Review* ini tidak hanya dibutuhkan oleh konsumen, tapi juga

pemilik restoran. Pemilik restoran dapat melihat bagaimana respon dari konsumen. Berikut *rating* dari *review* pada aplikasi Zomato pada Gambar 2.1 dan Gambar 2.2

Gambar 2.1 *Review* restoran Woodpecker *Coffee*

Gambar 2.2 *Review* Restoran Kopi Nalar

2.3 *Pre-Processing*

Pre-processing data adalah proses pembersihan dan mempersiapkan teks untuk klasifikasi (Haddi, et al., 2013). Pada *text mining* proses *pre-processing* dengan menghilangkan kata-kata yang tidak penting atau dapat disebut tidak memiliki arti dari dokumen tersebut. Dengan dihilangkannya kata tidak penting maka meringankan proses tersebut untuk menjadikan informasi pada dokumen. Proses ini dilakukan setelah pelabelan data dimana data disiapkan agar menjadi data yang siap dianalisis ada beberapa tahap dalam *pre-processing* yang meliputi *case folding, filtering*, tokenisasi, *stopword removal* dan *stemming*. Hasil dari *pre-*

processing ini akan dilanjutkan untuk memberikan bobot setiap kata dan akan dilanjutkan untuk klasifikasi *rating*.

2.3.1 *Case Folding*

Proses pertama dari *review* yang berhuruf kapital menjadi huruf kecil (*lower case*) pada teks tersebut disebut *case folding*. Berikut contoh mengenai *case folding* pada Tabel 2.2.

Tabel 2.2 *Case Folding*

Masukkan dari *Review* Restoran	Output
Saya pesan waffle dan rasanya asin sekali, pelayanannya gak bagus	saya pesan waffle dan rasanya asin sekali, pelayanannya gak bagus

2.3.2 *Filtering*

Filtering merupakan hasil dari proses *case folding*, *filtering* berguna untuk menghilangkan tanda baca dan angka yang terdapat dalam sebuah kalimat. Berikut contoh dari filtering pada Tabel 2.3.

Tabel 2.3 *Filtering*

Masukkan dari *Review* Restoran	Output
saya pesan waffle dan rasanya asin sekali, pelayanannya gak bagus	saya pesan waffle dan rasanya asin sekali pelayanannya gak bagus

2.3.3 Tokenisasi

Proses pemecah dokumen menjadi beberapa kumpulan kata yang berasal dari proses *filtering*. Tokenisasi dilakukan untuk memisahkan per spasi. Berikut Tabel 2.4 mengenai contoh tokenisasi.

Tabel 2.4 Tokenisasi

Masukkan dari *Review* Restoran	Output
saya pesan waffle dan rasanya asin sekali	saya pesan

Masukkan dari *Review* Restoran	Output
pelayanannya gak bagus	waffle dan rasanya asin sekali pelayanannya gak bagus

2.3.4 *Stopword Removal*

Proses *stopword removal* merupakan hasil dari proses tokenisasi yaitu dengan menghilangkan kata-kata yang tidak penting atau tidak memiliki arti, berikut kata-kata yang dihilangkan dalam teks seperti “dan”,“yang”,“di”,“ke”,”dari” dan lain sebagainya yang terdapat dalam *stopword*. Dihilangkannya kata-kata *stopword* dikarenakan penggunaannya terlalu umum, dan lebih berfokus kepada kata-kata penting (Ganesan, 2015). Stopword yang digunakan dalam penelitian ini berupa *stopword* pada data *mining*. Berikut contoh proses *stopword removal* pada Tabel 2.5.

Tabel 2.5 *Stopword Removal*

Masukkan dari *Review* Restoran	Output
saya pesan waffle dan rasanya asin sekali pelayanannya gak bagus	pesan waffle rasanya asin pelayanannya gak bagus

2.3.5 *Stemming*

Proses stemming yaitu dengan menghilangkan imbuhan yang terdapat dalam kata. Proses *stemming* merupakan pencarian kata dasar yang berbeda-beda dan memiliki beberapa jenis *stemming.* Penelitian ini menggunakan *stemming* sastrawi serta menggunakan *library* sastrawi yang sederhana untuk digunakan secara mudah dan begitupula dokumentasinya (Librian, 2017). Algoritme *stemming* merupakan algoritme berbasis Nazief dan Adriani, kemudian

ditingkatkan oleh algortime CS (*Confix Stripping*), kemudian ditingkatkan lagi oleh algoritme ECS (*Enhanced Confix Stripping*), lalu ditingkatkan lagi oleh *Modified* ECS. Berikut contoh proses *stemming* pada Tabel 2.6.

Tabel 2.6 *Stemming*

Masukkan dari *Review* Restoran	Output
pesan	pesan
waffle	waffle
rasanya	rasa
asin	asin
pelayanannya	layan
gak	gak
bagus	bagus

2.4 Pembobotan TF

Term Frequency (TF) merupakan pengukuran yang paling sederhana dalam metode pembobotan. Pada metode ini masing-masing *term* diasumsikan mempunyai proporsi kepentingan sesuai jumlah terjadinya dalam teks dokumen. *Term* merupakan kata yang banyak muncul dalam dokumen yang memiliki nilai bobot yang tinggi dengan pendekatan *bag of words* yaitu dengan mengabaikan urutan kata secara sintaksis hanya menekan pada menghitung kata pada dokumen (Manning, et al,. 2009). Salah satu metode yang populer untuk melakukan pembobotan kata adalah TF-IDF (*Term Frequency-Inverse Document Frequency*). *Term Frequency-Inverse Document Frequency*) adalah sebuah metode pembobotan yang menggabungkan dua konsep yaitu *Term Frequency* dan *Document Frequency* yang bertujuan untuk mengindikasikan seberapa unik terhadap kemunculan kata pada kumpulan dokumen (Nurjannah, 2016). Berikut Persamaan dari *Term Frequency-Inverse Document Frequency.*

Term Frequency (TF) merupakan banyaknya kemunculan *term*/token/kata pada dokumen. Nilai TF dapat menggunakan Persamaan 2.1.

$$tf_{t,d} = \begin{cases} 1 + log10\ tf_{t,d} & if\ tf_{t,d} > 0 \\ 0 & otherwise \end{cases} \quad (2.1)$$

Setelah menghitung *Term frequency* maka selanjutnya dibutuhkan perhitungan *document frequency* yang bertujuan untuk mengetahui jumlah dokumen setiap *term* pada Persamaan 2.2.

$$idf_t = log10\frac{N}{df_t} \quad (2.2)$$

Setelah mendapatkan nilai *term frequency* dan idf_t. Maka selanjutnya menghitung nilai TF-IDF menggunakan Persamaan 2.3.

$$W_{t,d} = tf_{t,d}\ x\ idf_t \quad (2.3)$$

Keterangan:

$tf_{t,d}$ = Frekuensi kemunculan kata t pada dokumen d

idf_t = Banyak dokumen yang mengandung term/token/kata

N = Jumlah dokumen

df_t = Banyak dokumen yang memuat t

$W_{t,d}$ = Bobot suatu *term*/token/kata

2.5 Jaringan Saraf Tiruan

Menurut Jatmiko, jaringan syaraf tiruan dibuat dan dikembangkan untuk lebih mengenal organ otak dan mengambil keunggulannya dari otak. JST dikembangan dari beberapa tahun yang lalu. Kemajuan dalam bidang neurobiology memungkinkan para peneliti untuk membangun model-model matematika dari sel-sel syaraf untuk mensimulasikan perilaku jaringan syaraf manusia.

Jaringan syaraf tiruan (JST) merupakan sistem pemrosesan informasi yang memiliki karakteristik serupa dengan jaringan neural biologis. Ada berbagai jenis JST yang telah diciptakan dan ada tiga hal yang akan mencirikan setiap jenis JST, yakni:

- Pola hubungan/koneksi antara elemen-elemen sederhananya, yakni neuron
- Metode penentuan nobot koneksi antar *neuron*
- Fungsi Aktivasinya

2.6 Normalisasi

Normalisasi dilakukan ketika nilai input memiliki jarak yang bervariasi seperti puluhan hingga ribuan. Normalisasi sangat berguna untuk klasifikasi algoritme yang melibatkan jaringan saraf, atau jarak pengukuran seperti klasifikasi tetangga terdekat dan pengelompokan. Ada berbagai metode dalam melakukan normalisasi seperti *Z-Score, Decimal Scaling,* dan *Min-Max Normalization (*Jain & Bhandare, 2011). Pada penelitian ini menggunakan *Min-Max Normalization.* Berikut Persamaan 2.4 yaitu *Min-Max Normalization.*

$$d' = \frac{d - \min(p)}{\max(p) - \min(p)} \quad (2.4)$$

Keterangan:

d' = hasil normalisasi

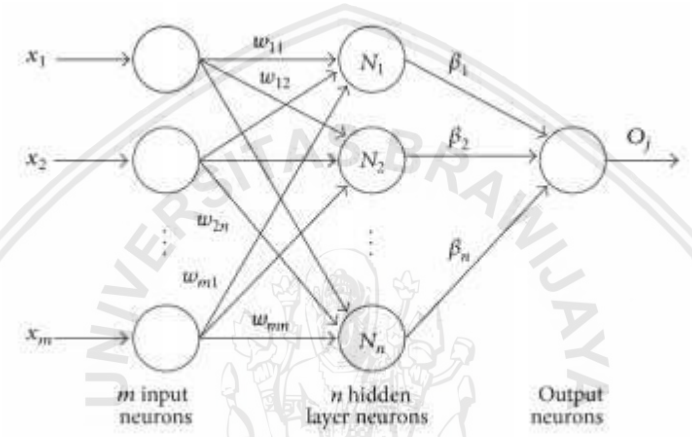
d = data yang dilakukan normalisasi

$\min(p)$ = nilai minimum yang terdapat dalam data

$\max(p)$ = nilai maksimum yang terdapat dalam data

2.7 *Extreme Learning Machine*

Extreme Learning Machine merupakan jaringan syaraf tiruan *feedforward* dengan satu *hidden layer* atau biasa disebut dengan *istilah single hidden layer feedforward neural network* (Huang, et al., 2006). Metode ini dibuat untuk mengatasi kelemahan-kelemahan dari jaringan syaraf tiruan *feedforward* terutama dalam hal *learning speed*. Berikut arsitektur ELM pada Gambar 2.3.



Gambar 2.3 Arsitektur ELM

Sumber: You et al. (2014)

2.7.1 Training *Extreme Learning Machine*

Langkah-langkah yang dilakukan pada data *training* menggunakan metode *Extreme Learning Machine* diantaranya:

1. Hitung *input weight* dan bias dengan *random*
2. Hitung inisialisasi *output hidden layer*

$$H_{init} = X\ training\ .\ W^{\mathrm{T}} + b \tag{2.5}$$

Keterangan:

H_{init} = Hasil *output hidden layer*

$X\ training$ = Data *training*

W^{T} = Transpose nilai *input weight*

b = Nilai bias

3. Hitung Aktivasi menggunakan Sigmoid Biner pada Persamaan 2.11.
4. Hitung *Moore Penrose*

$$H^{+} = (H^{\mathrm{T}} .\ H)^{-1}.H^{\mathrm{T}} \quad (2.6)$$

Keterangan:

H^{+} = Matriks *Moore Penroose*

H = Matriks *output hidden layer* dengan fungsi aktivasi

H^{T} = Matriks transpose *output hidden layer* dengan fungsi aktivasi

$(H^{\mathrm{T}} .\ H)^{-1}$ = Matriks inverse perkalian matriks transpose *output hidden layer* dengan fungsi aktivasi dengan *output hidden layer* dengan fungsi aktivasi

5. Menghitung *Output Weight*

$$\beta = H^{+} .\ Y \quad (2.7)$$

Keterangan:

β = Nilai *output weight*

H^{+} = Nilai matriks *moore penrose*

Y = Nilai matriks target

2.7.2 Testing *Extreme Learning Machine*

Langkah-langkah yang dilakukan pada data *testing* menggunakan metode *Extreme Learning Machine* diantaranya:

1. Didapatkan *input weight*, bias, dan β dari proses *training* sebelumnya
2. Hitung Inisialisasi *output hidden layer* menggunakan Persamaan 2.8 sebagai berikut:

$$H_{init} = X\ testing\ .W^{\mathrm{T}} + b \quad (2.8)$$

Keterangan:

H_{init} = Hasil *output hidden layer*

$X\ testing$ = Data *testing*

W^{T} = Transpose nilai *input weight*

b = Nilai bias

3. Hitung matriks pada Aktivasi Sigmoid Biner menggunakan Persamaan 2.11 seperti yang dilakukan pada data *training*

4. Hitung hasil persamaan prediksi (ŷ)

$$\hat{y} = H \,.\, \beta \tag{2.9}$$

Keterangan:

$\hat{y}$ = Nilai *output layer*

H = Nilai *output hidden layer* dengan fungsi aktivasi

β = Nilai *output weight* pada proses *training*

5. Hitung evaluasi menggunakan akurasi

$$Akurasi = \frac{Data\ uji\ yang\ benar}{Semua\ data}\ x\ 100\% \tag{2.10}$$

2.8 Fungsi Aktivasi

Pada fungsi aktivasi di algoritme *Extreme Learning Machine* adalah suatu fungsi yang memetakan penjumlahan input elemen pemroses terhadap outputnya. Fungsi aktivasi yang diterapkan menurut (Srimuang, et al,. 2015) yaitu diantaranya:

1. Fungsi Aktivasi Sigmoid Biner

$$h(x) = \frac{1}{1+\exp(-Hinit)} \tag{2.11}$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$\exp(-Hinit)$ = Exponensial minus nilai Hinit

2. Fungsi Aktivasi Sin

$$h(x) = \sin(H_{init}) \tag{2.12}$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$\sin(H_{init})$ = Nilai sin dari Hinit

3. Fungsi Aktivasi Sigmoid Bipolar

$$h(x) = \frac{1-\exp(-Hinit)}{1+\exp(-Hinit)} \tag{2.13}$$

Keterangan:

$h(x)$ = Fungsi aktivasi

$\exp(-Hinit)$ = Exponensial minus nilai Hinit

2.9 *K-Fold Cross Validation*

K-Fold Cross Validation merupakan teknik validasi untuk mengetahui kesesuaian data dengan sistem tersebut. Pada teknik *K-Fold Cross Validation*, data dibagi menjadi *k* bagian masing-masing *k* bagian memiliki jumlah yang sama. Misalkan menggunakan 5 *fold*, pertama menggunakan data *testing* pada *fold* pertama dan sisanya data latih. Selanjutnya berulang hingga *fold* terakhir, tujuan dari teknik ini untuk memberikan kesempatan dalam melakukan uji terhadap dataset (Kohavi, 2015). Pada Gambar 2.4 ditunjukkan sebagai ilustrasi *K-Fold Cross Validation*.

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

Data Uji

Data Latih

Gambar 2.4 Ilustrasi *K-fold cross validation* = 5

BAB 3 METODOLOGI PENELITIAN

Pada bab ini menunjukkan secara terperinci mengenai metodologi ataupun tahapan-tahapan yang dilakukan untuk penelitian dalam mengatasi masalah Prediksi *Rating* Otomatis berdasarkan *Review* Restoran pada Aplikasi Zomato dengan menggunakan metode *Extreme Learning Machine* (ELM). Pembahasan yang akan dijelaskan yaitu tipe penelitian yang digunakan, strategi penelitian berupa studi kasus, partisipan penelitian, lokasi yang digunakan untuk penelitian, data penelitian, dan implementasi algoritme.

3.1 Tipe Penelitian

Penelitian non implementatif yang digunakan pada tipe penelitian ini merupakan penjelasan korelasi antar unsur dalam objek penelitian pada kondisi tertentu yang sedang diteliti. Hasil utama pada tipe penelitian analitik yaitu hasil dari analisis. Jenis yang digunakan merupakan jenis kuantitatif karena menggunakan data *review* restoran dari aplikasi Zomato dari berbagai restoran untuk memprediksi secara otomatis *rating* dari restoran tersebut berdasarkan *review.*

3.2 Strategi Penelitian

Dalam memgembangkan penelitian ini dibutuhkan berupa permasalahan yang berfungsi untuk menganalisis pengaruh *rating* yang terdapat pada aplikasi Zomato pada restoran di Jakarta. Selain itu strategi penelitian dengan memilih permasalahan pada restoran tersebut yang ditujukan untuk mengetahui pengaruh pada *review* pada restoran tersebut.

3.3 Partisipan Penelitian

Partisipan yang terlibat dalam penelitian ini yaitu konsumen restoran berdasarkan *review* pada aplikasi Zomato. Alasan pemilihan partisipan tersebut untuk mengetahui permasalahan dan hanya untuk validasi data pada *review* yang terdapat *rating* restoran pada aplikasi Zomato tersebut.

3.4 Lokasi Penelitian

Penelitian ini dilakukan di Laboratorium Riset Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. Lokasi yang dilakukan oleh peneliti adalah *review* dari restoran yang berada di Jakarta.

3.5 Teknik Pengumpulan Data

Pengumpulan data dilakukan menggunakan data primer yaitu didapatkan langsung dari *review* restoran pada aplikasi Zomato dan juga dapat dibuka melalui link https://www.zomato.com/id/jakarta. Teknik pengumpulan data tersebut juga diperoleh dari berbagai restoran di Jakarta.

3.6 Data Penelitian

Dalam data penelitian dari *review* Restoran pada aplikasi Zomato. Data tersebut meliputi data *review* beserta *rating* sebanyak 150 data yang digunakan. Jumlah rating dalam data latih merupakan penjabaran dari masing-masing *rating* yaitu *rating* 1 hingga *rating* 5 dengan jumlah yang sama pada masing-masing *rating*.

3.7 Teknik Analisis Hasil

Teknik analisis hasil dilakukan melalui proses pengujian dimana data *review* tersebut akan dilakukan *pre-processing* yang meliputi *Case Folding*, *Filtering*, Tokenisasi, *Stopword Removal* dan *Stemming* setelah itu akan dilakukan pembobotan setiap kata dengan menggunakan *Term Frequency-Inverse Document Frequency* (TF-IDF) agar dapat diklasifikasikan kedalam *rating*. Setelah pembobotan kata dilakukan normalisasi dan masuk perhitungan *Extreme Learning Machine* (ELM). Perhitungan terdapat data latih dan data uji. Data uji dilakukan setelah mendapatkan nilai dari data latih.

3.8 Implementasi Algoritme

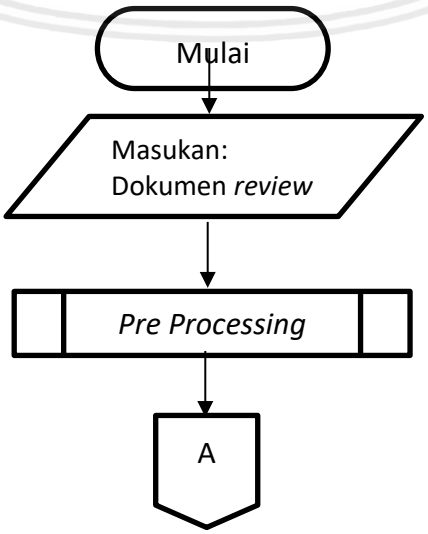
Tahap implementasi algoritme menggunakan *Extreme Learning Machine* (ELM) adalah dilakukan perhitungan manualisasi untuk mempermudah peneliti mengetahui langkah perhitungan ELM. Selain itu untuk membandingkan antara hasil perhitungan manualisasi maupun dengan pengujian metode. Tahap selanjutnya dengan proses pembagian data *training* dan data *testing*, selanjutnya untuk melakukan evaluasi menggunakan akurasi.

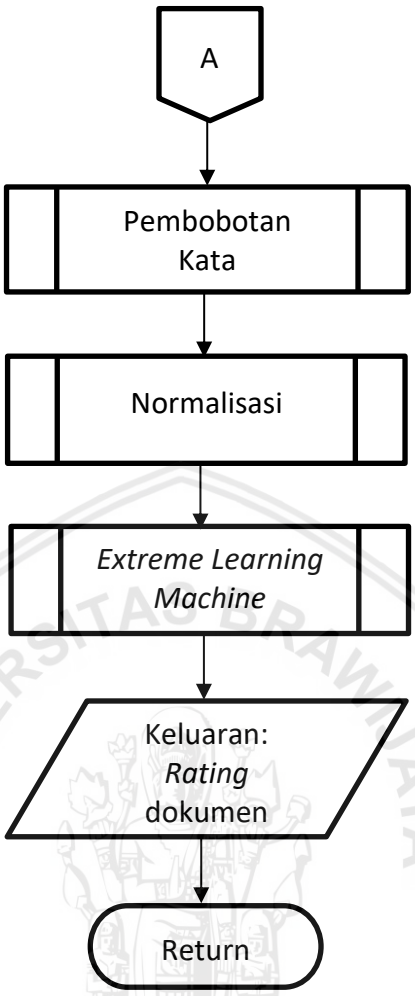
BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Pada bab ini akan dilakukan perancangan untuk menyelesaikan permasalahan penelitian terkait yaitu prediksi *rating* otomatis berdasarkan *review* restoran menggunakan metode *Extreme Learning Machine* (ELM). Hal ini dilakukan untuk mencapai tujuan penelitian ini yaitu dapat memprediksi *rating* berdasarkan *review* yang ditulis oleh konsumen pada aplikasi tersebut. Metode *Extreme Leraning Machine* mampu menghasilkan prediksi dengan nilai akurasi yang tinggi dari penelitian-penelitian sebelumnya, selain itu mampu menghasilkan *learning speed* yang cukup baik

Pengujian metode ini terdiri dari masukan berupa data *review* restoran dalam aplikasi Zomato. Terdapat parameter perhitungan yaitu jumlah data *training* dan data *testing* menggunakan *k-fold cross validation*, banyaknya jumlah *neuron* pada *hidden layer*, serta fungsi aktivasi yang digunakan. Dalam penelitian ini melakukan *pre-processing* terlebih dahulu yaitu *case folding, filtering,* tokenisasi, *stopword removal*, dan *stemming*. Dalam pengujian metode ini dapat memprediksi *rating* 1 hingga *rating* 5. Proses perhitungannya, ketika *pre-processing* selesai maka dilakukan pembobotan kata menggunakan TF-IDF, setelah selesai maka data tersebut di normalisasi, setelah normalisasi selesai maka selanjutnya proses ELM, pada proses ini membagi data menjadi *training* dan *testing* dengan jumlah yang ideal. Untuk proses awal pada *training* menggunakan *input weight* dan bias dengan *random* selanjutnya melakukan perhitungan *output hidden layer* menggunakan fungsi aktivasi. Setelah proses *training* selesai maka pada proses *testing* dimana dilanjutkan untuk perhitungan *output weight* yang akan digunakan pada tahap *testing* untuk menghitung prediksi yang dihasilkan. Berikut diagram alir prediksi *rating* berdasarkan *review*, untuk diagram alir permasalahan lebih detail digambarkan pada Gambar 4.1

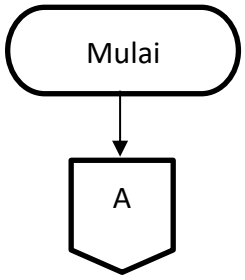


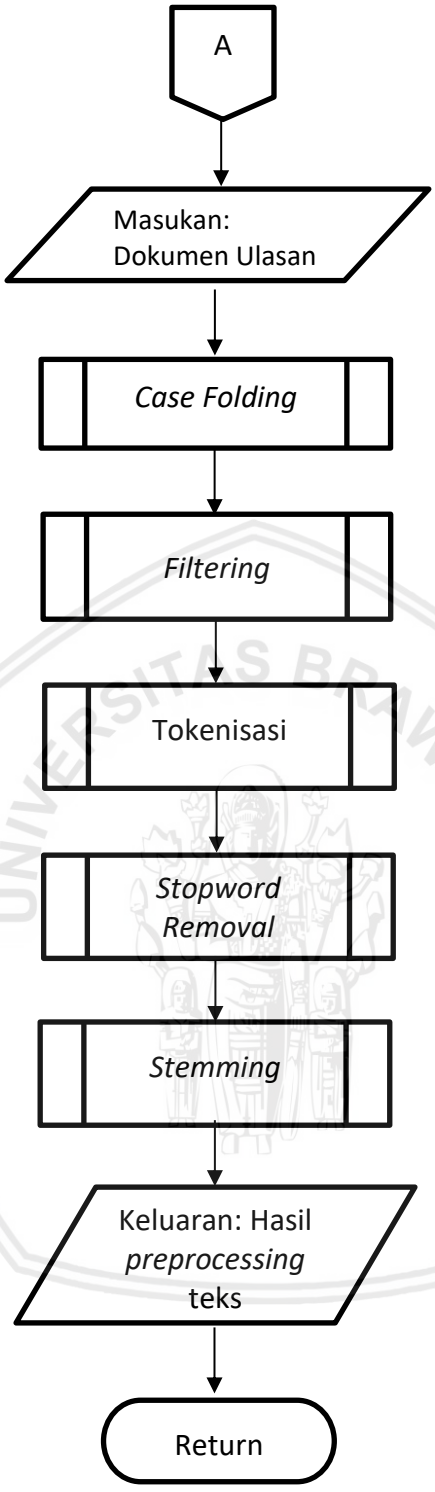


Gambar 4.1 Diagram alir formulasi permasalahan

4.2 Alur Proses *Pre-Processing*

Pada proses *pre-processing* teridiri dari case *folding, filtering,* tokenisasi, *stopword removal*, dan *stemming*. Hasil dari *pre-processing* akan dilakukan pembobotan kata dalam klasifikasi *rating* berdasarkan *review* restoran. Tahapan ini ditunjukkan pada Gambar 4.2.

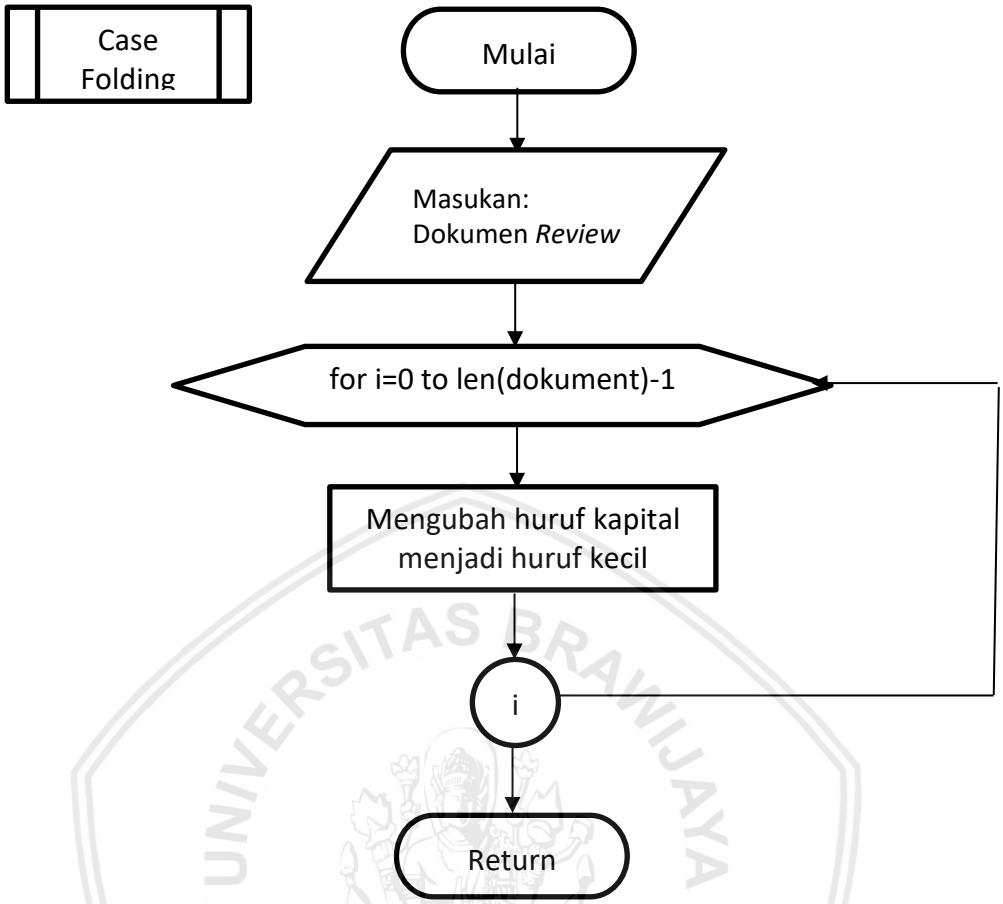




Gambar 4.2 Diagram alir *pre-processing*

4.2.1 *Case Folding*

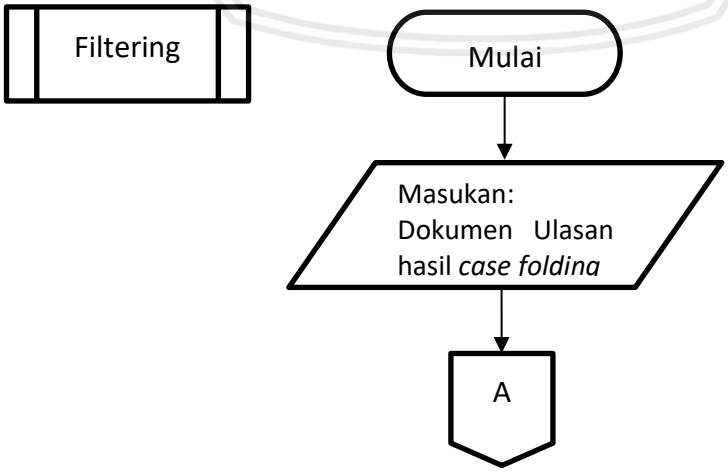
Case folding merupakan bagian awal dari *pre-processing* yang berperan untuk mengganti huruf kapital menjadi huruf kecil (*lowercase*). Tahapan ini ditunjukkan pada Gambar 4.3.

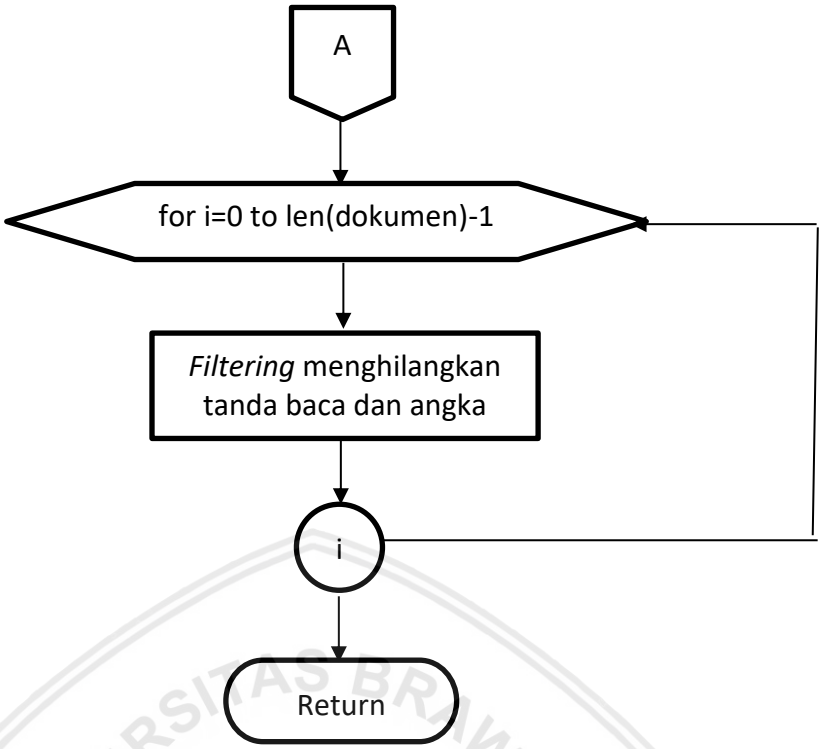


Gambar 4.3 Diagram alir *case folding*

4.2.2 *Filtering*

Filterig merupakan proses menghilangkan tanda baca dan angka pada hasil dari *case folding*. Tahapan ini ditunjukkan pada Gambar 4.4.

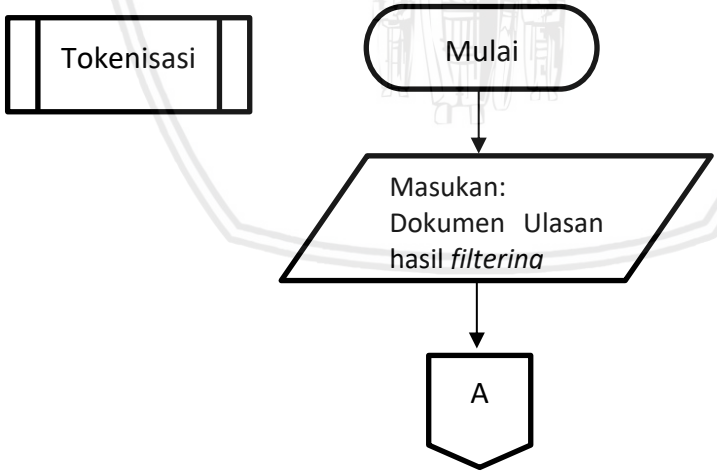


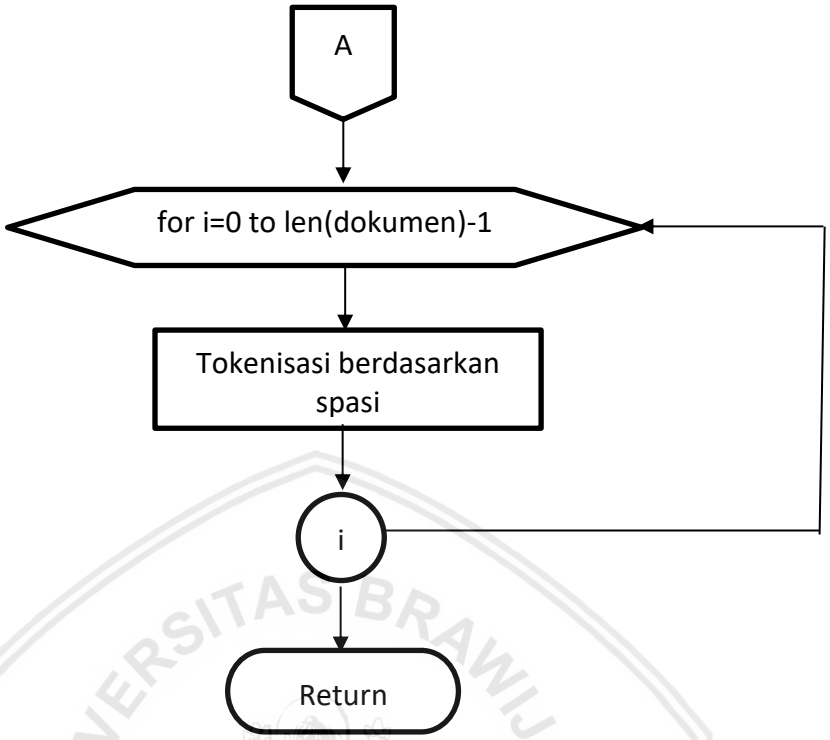


Gambar 4.4 Diagram alir *filtering*

4.2.3 Tokenisasi

Tokenisasi termasuk bagian dari *pre-processing* yang berguna untuk memecah kata menggunakan spasi setelah proses *filtering*. Tahapan ini ditunjukkan pada Gambar 4.5.

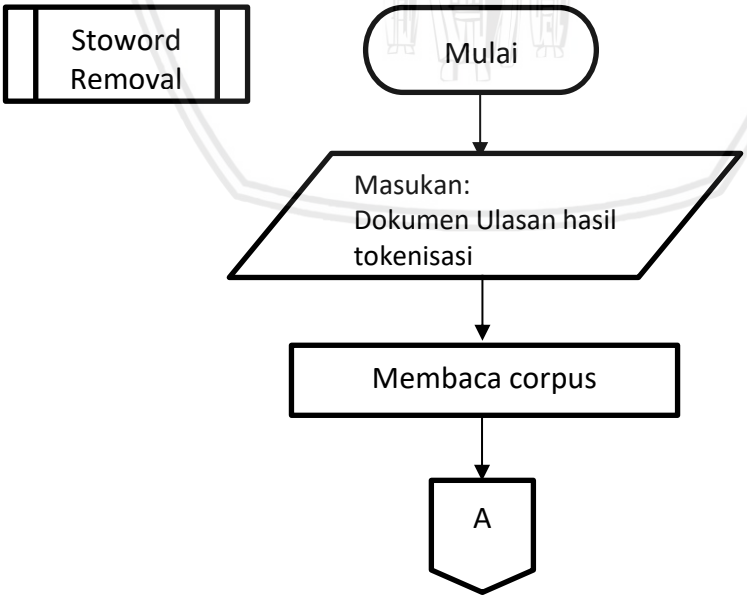


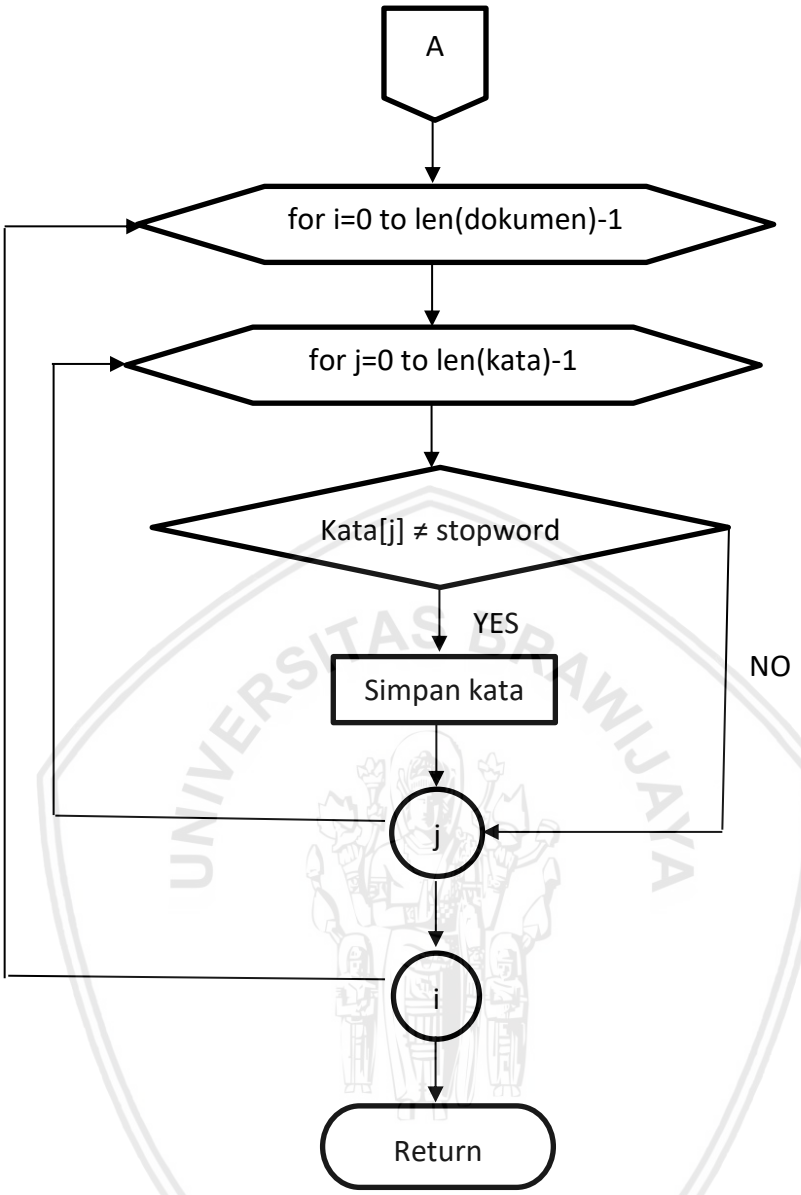


Gambar 4.5 Diagram alir tokenisasi

4.2.4 *Stopword Removal*

Stoword Removal merupakan bagian dari *pre-processing* yang berguna untuk menghilangkan kata-kata tidak penting. Tahapan ini ditunjukkan pada Gambar 4.6.

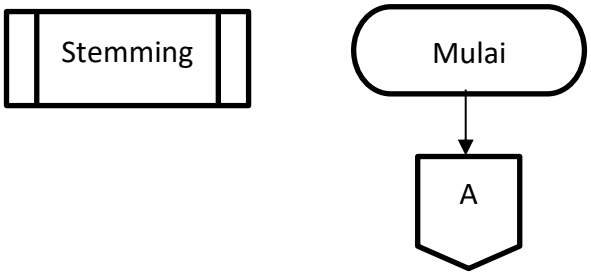


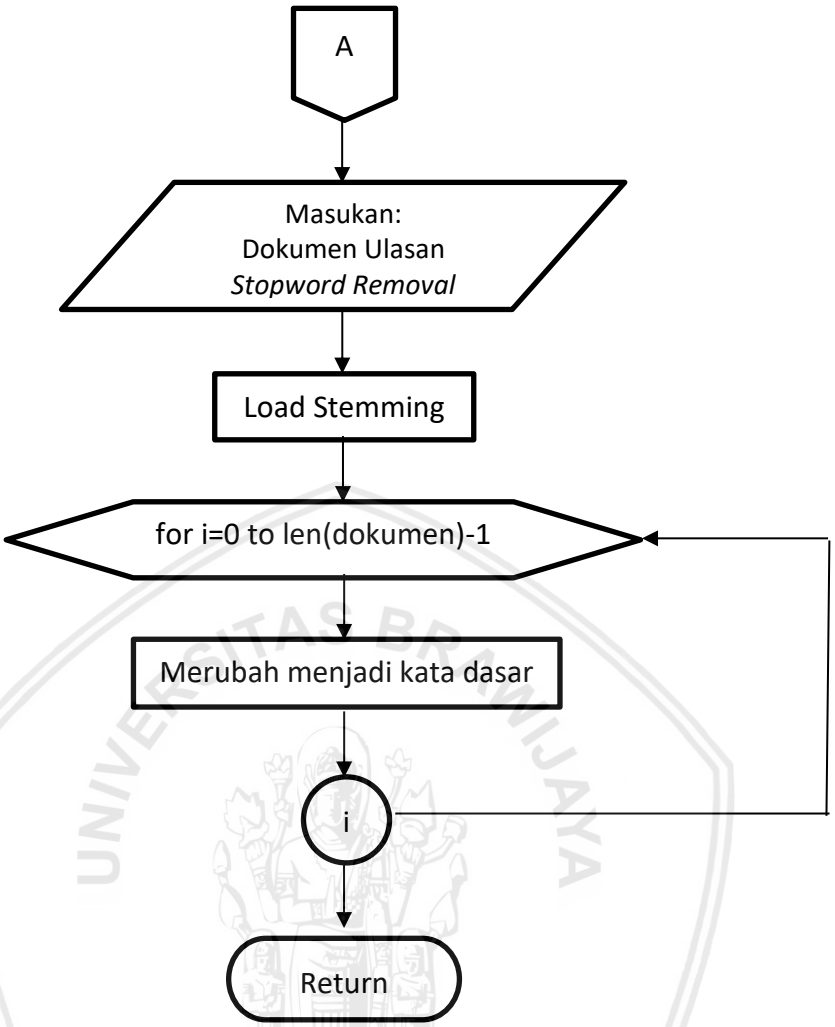


Gambar 4.6 Diagram alir *stopword removal*

4.2.5 *Stemming*

Stemming merupakan subbab pada *pre-processing* yang berguna untuk mengubah menjadi kata dasar. Tahapan ini lebih detailnya pada Gambar 4.7.

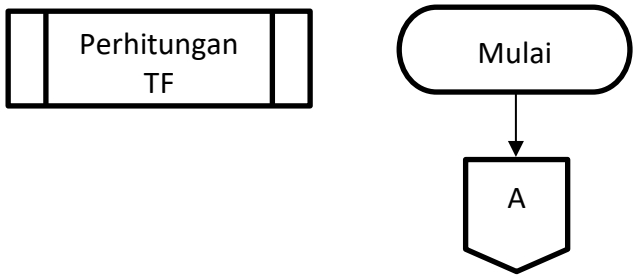


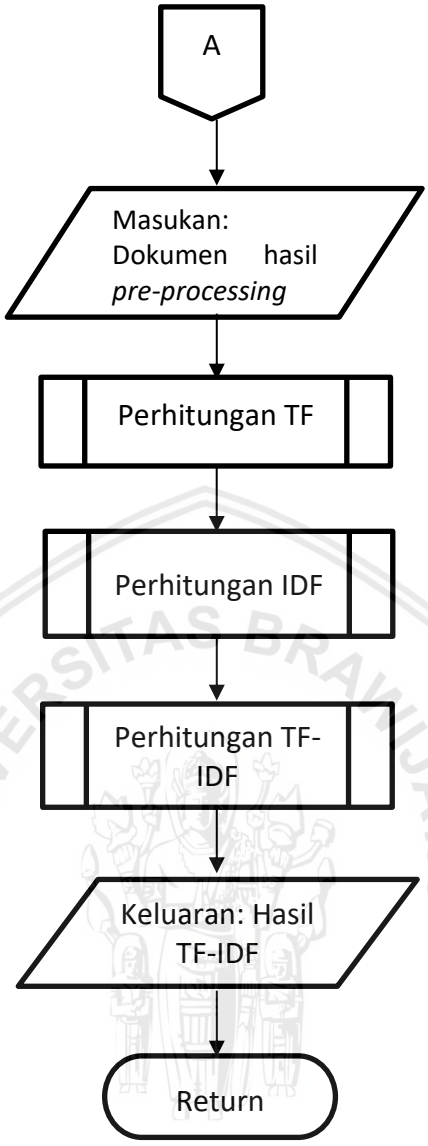


Gambar 4.7 Diagram alir *stemming*

4.3 Alur Proses Pembobotan Kata

Pada tahap ini merupakan proses pembobotan kata meliputi proses TF, IDF, dan TF-IDF. Hasil dari pembobotan kata menggunakan TF-IDF digunakan sebelum masuk perhitungan menggunakan metode *Extreme Learning Machine* untuk proses klasifikasi *rating* yang berguna untuk menentukan *rating* pada sebuah *review* restoran. Tahapan pembobotan kata akan ditunjukkan pada Gambar 4.8.

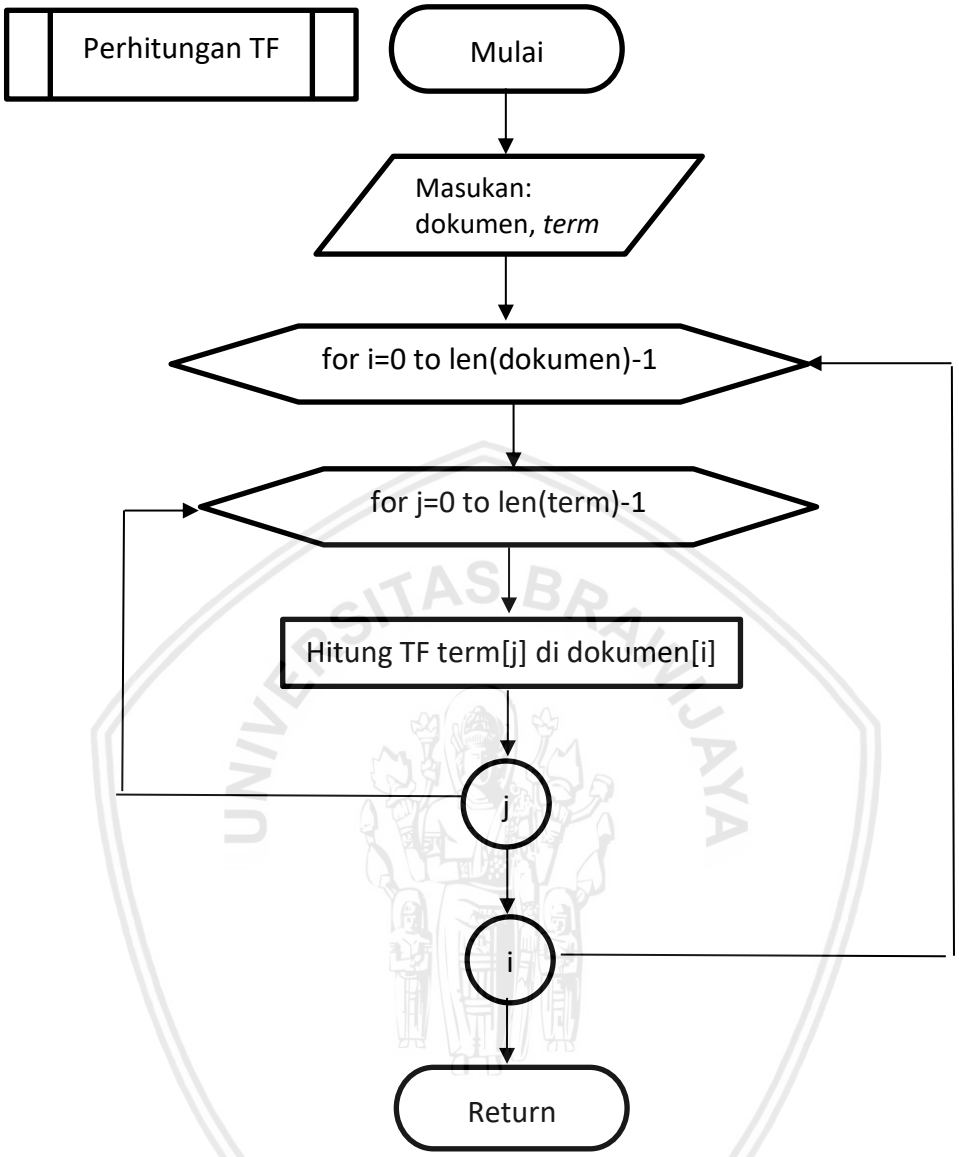




Gambar 4.8 Diagram alir pembobotan kata

4.3.1 Proses *Term Frequency*

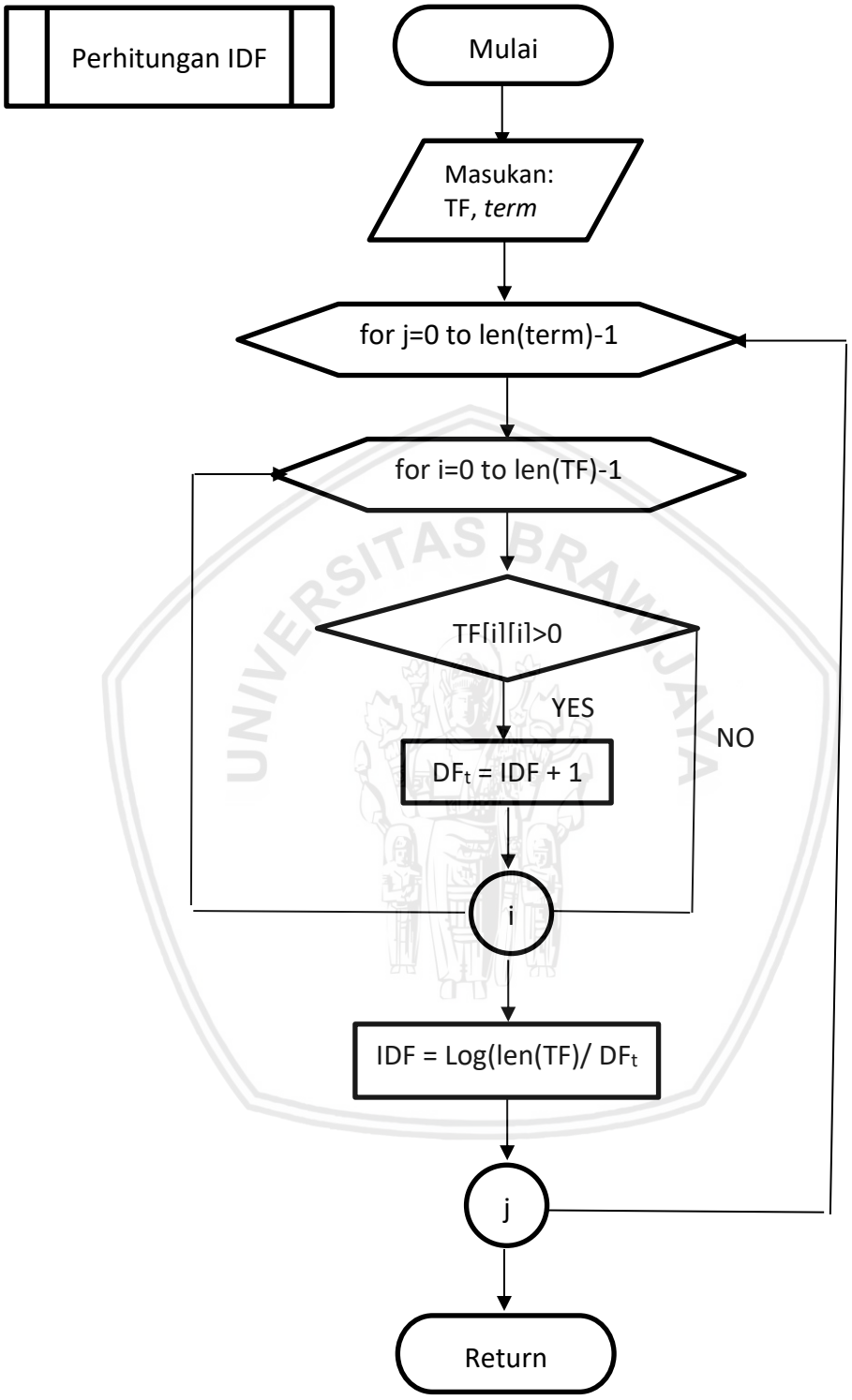
Term Frequency merupakan subbab dari pembobotan kata. Nilai TF merupakan frekuensi kata yang terdapat pada suatu dokumen. Proses *Term Frequency* akan ditunjukkan pada Gambar 4.9



Gambar 4.9 Diagram alir proses *Term Frequency*

4.3.2 Proses *Inverse Document Frequency*

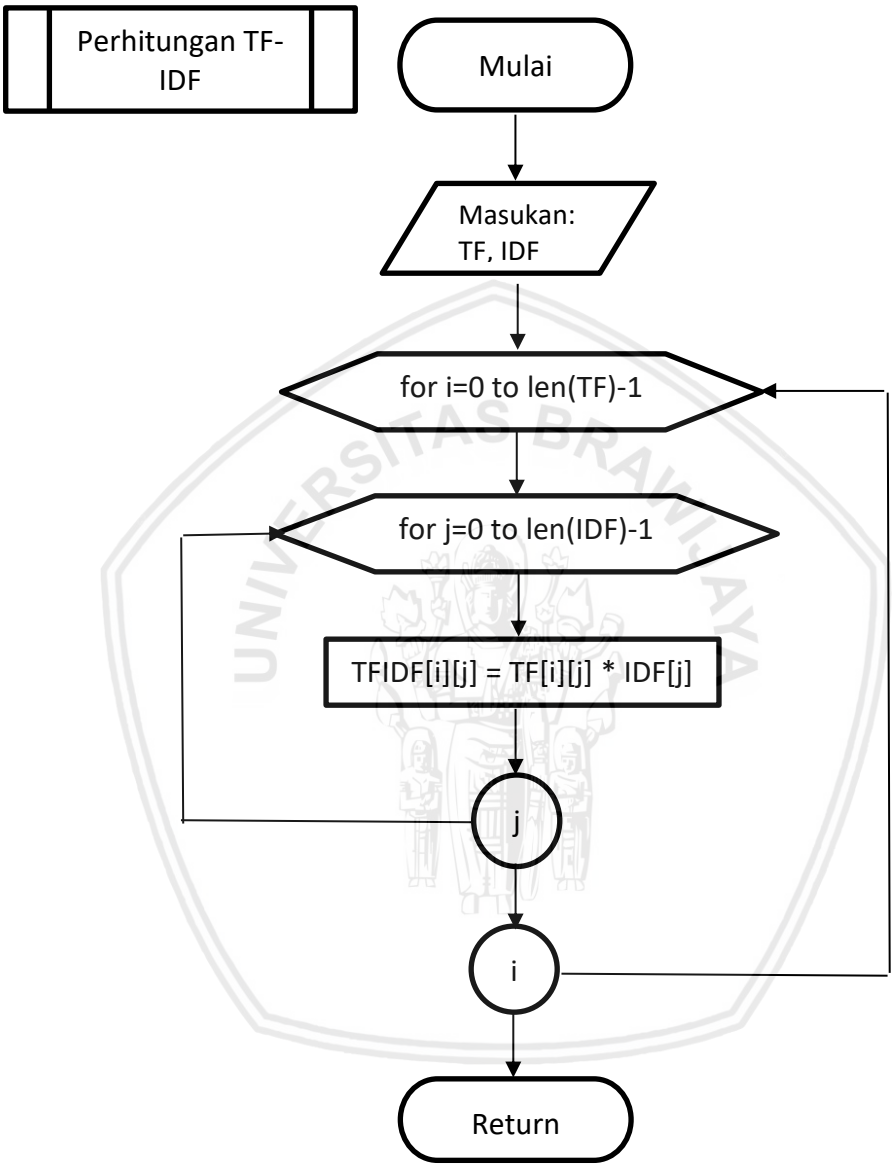
Proses *Inverse Document Frequency* merupakan bagian dari pembobotan kata pada TF-IDF. Kata yang jarang muncul memiliki nilai *invers document frequency* yang tinggi. Diagram alir IDF akan dijelaskan pada Gambar 4.10



Gambar 4.10 Diagram alir proses *Inverse Document Frequency*

4.3.3 Proses TF-IDF

TF-IDF merupakan subbab dari pembobotan kata. Nilai TF-IDF merupakan hasil perkalian dari *Term Frequency* dengan *Inverse Document Frequency*. Proses TF-IDF akan ditunjukkan pada Gambar 4.11



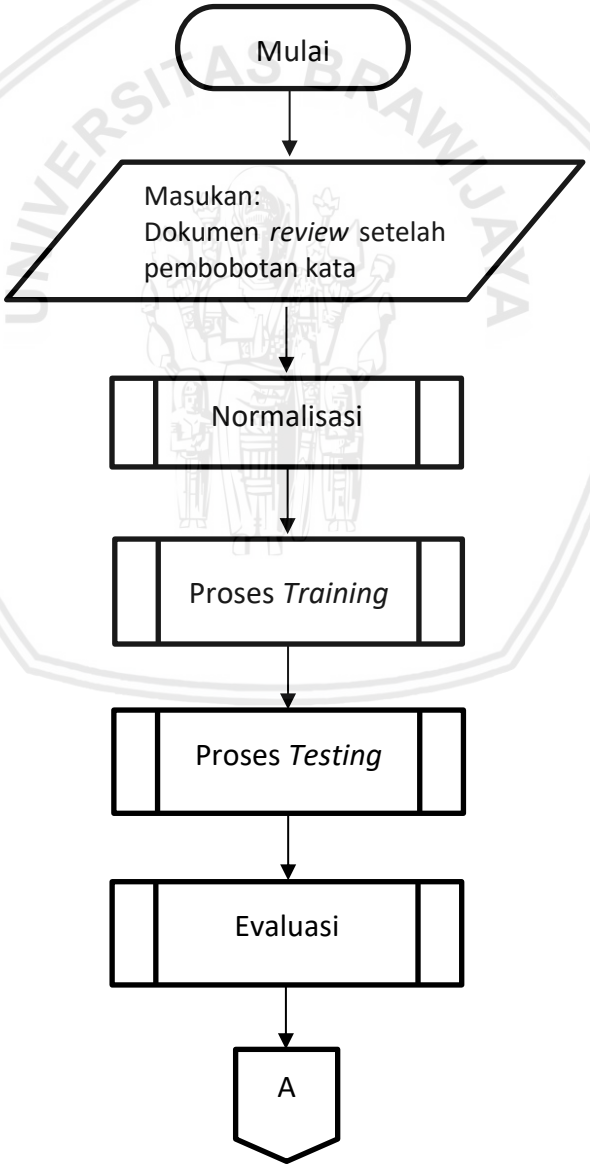
Gambar 4.11 Diagram alir proses TF-IDF

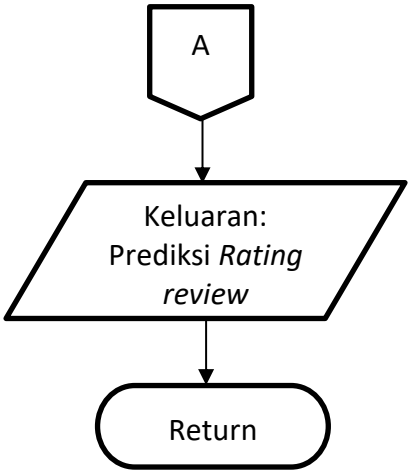
4.4 Proses Algoritme *Extreme Learning Machine*

Proses ini menjelaskan mengenai alur metode *Extreme Learning Machine*. Tahap pertama dimulai dari proses *training*, proses *testing*, dan evaluasi menggunakan akurasi. Pada Gambar 4.12 akan menunjukan alur dari metode hasil prediksi *rating* dengan metode *Extreme Learning Machine*.

Berikut diagram alir proses algoritme *Extreme Learning Machine* pada Gambar 4.12 diantaranya:

1. Masukkan berupa data *review* restoran pada aplikasi Zomato
2. Proses perhitungan data *training*, akan menghasilkan *output weight*
3. Proses perhitungan *testing* dilakukan untuk menghasilkan hasil prediksi sebelum masuk pada tahap evaluasi
4. Proses terakhir yaitu melakukan evaluasi menggunakan nilai akurasi

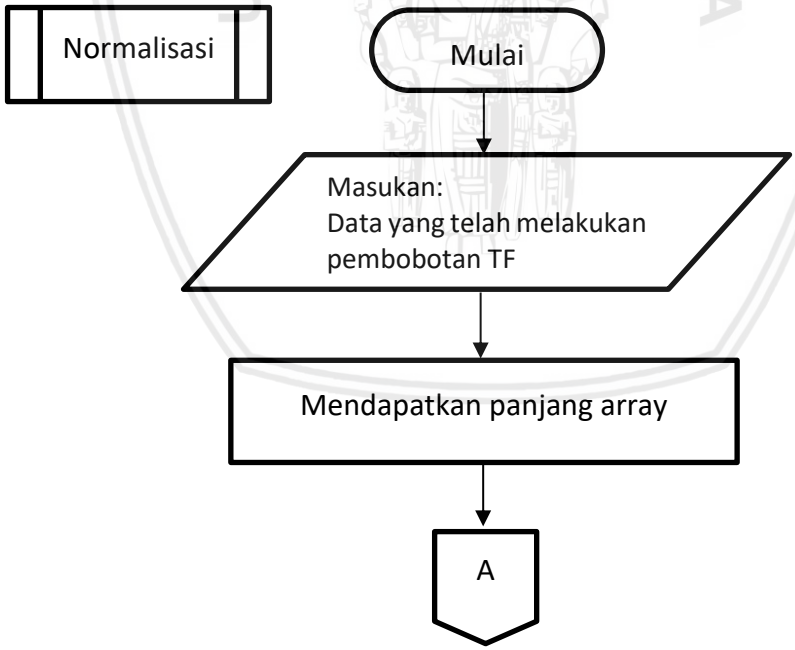


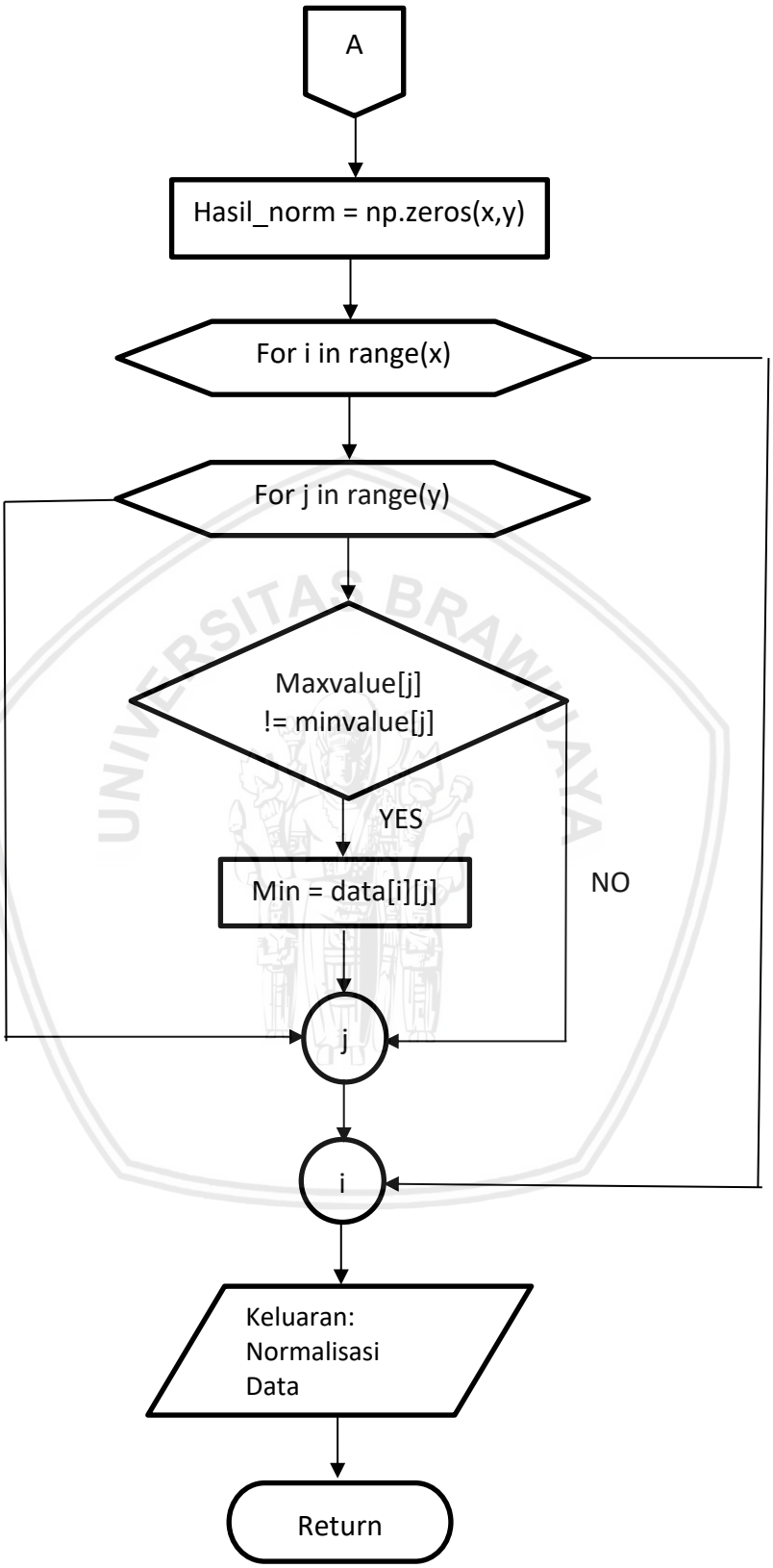


Gambar 4.12 Diagram alir metode *Extreme Learning Machine*

4.4.1 Normalisasi

Sebelum melakukan proses prediksi terhadap *review* restoran pada aplikasi Zomato, langkah pertama yang dilakukan sebelum proses *training* yaitu proses normalisasi dengan normalisasi *min-max*. Pada Gambar 4.13 ditunjukkan mengenai diagram alir normalisasi.

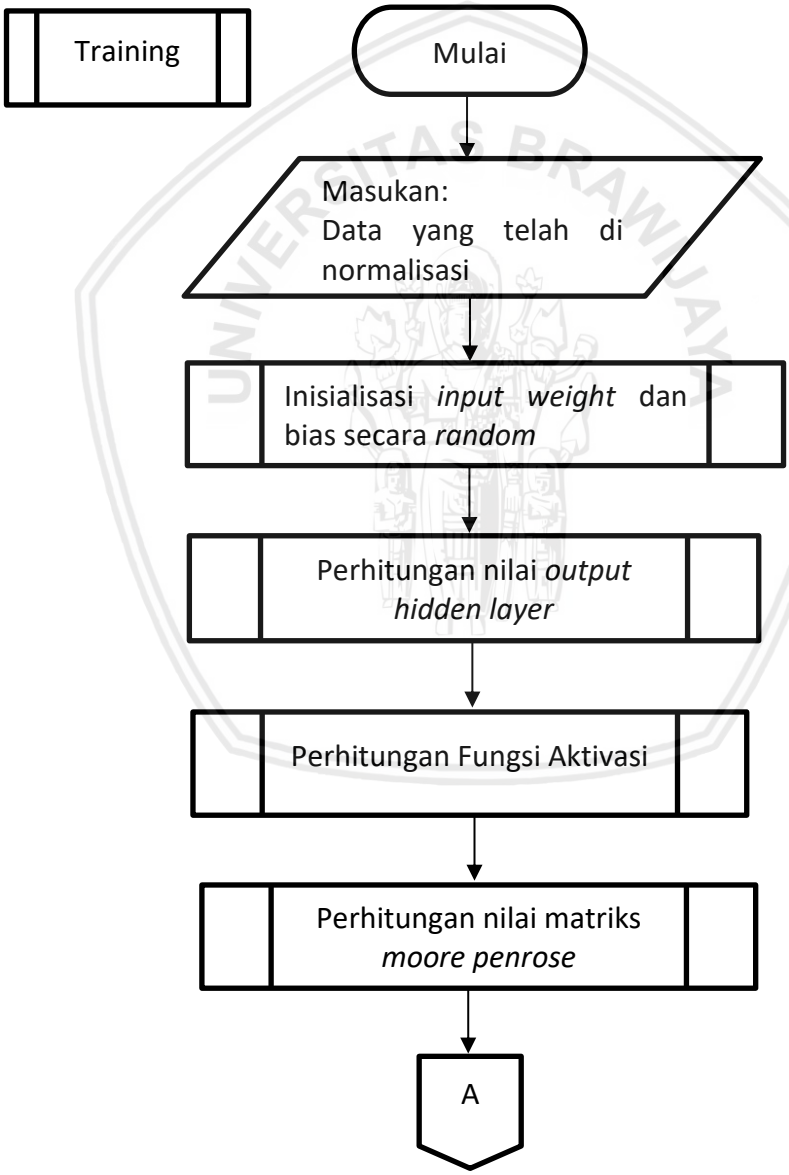


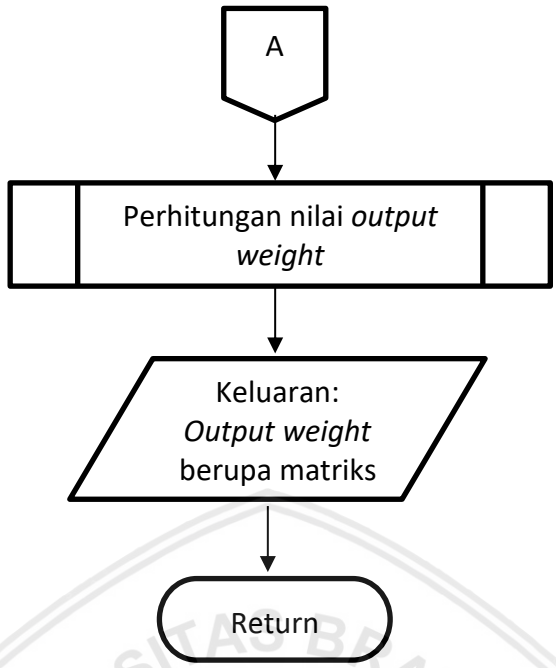


Gambar 4.13 Diagram alir proses *training*

4.4.2 Proses *Training*

Setelah proses *pre-processing*, pembobotan kata, dan normalisasi telah selesai maka masuk pada proses *training* metode *Extreme Learning Machine* (ELM). Proses training ini dilakukan untuk mendapatkan nilai *output weight* pada tahap akhir proses *training* untuk dilanjutkan ke proses *testing*. Dalam proses ini data yang digunakan berupa *review* restoran, selanjutnya menghitung bobot dan bias secara random untuk menghitung *output hidden layer*, setelah itu menghitung menggunakan fungsi aktivasi, setelah itu masuk tahap perhitungan *moore-penrose* yaitu perhitungan menggunakan invers dari OBE, dan selanjutnya menghitung nilai *output weight* dengan menghasilkan nilai prediksi. Berikut gambaran secara umum proses *training* pada Gambar 4.14





Gambar 4.14 Diagram alir proses *training*

Sesuai dengan Gambar 4.14, berikut penjelasan dari langkah-langkah proses *training* diantaranya:

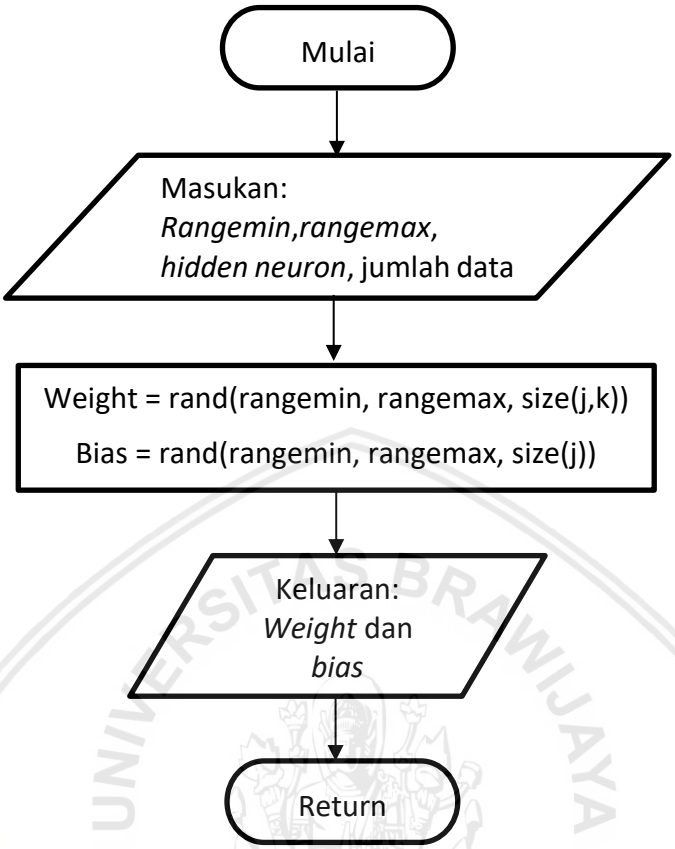
1. Inisialisasi *input weight* dan bias secara *random*
2. Menghitung *output hidden layer* menggunakan *input weight* dan bias
3. Menghitung menggunakan fungsi aktivasi Sigmoid Biner
4. Menghitung matriks *moore-penrose* berdasarkan persamaan 2.6
5. Menghitung *output weight* berdasarkan persamaan 2.7 dari hasil tersebut akan digunakan untuk perhitungan proses *testing*.
6. Mendapatkan nilai *output weight* untuk proses *testing*

4.4.2.1 Inisialisasi Input Weight dan Bias

Inisialisasi pada *input weight* serta bias dilakukan secara *random* dengan interval sebesar -1 sampai 1. Nilai *input weight* serta bias yang dihasilkan maka akan disesuaikan dengan jumlah *neuron* pada *hidden layer*. Pada Gambar 4.15 akan dijelaskan secara detail mengenai *input weight* serta bias.

Sesuai dengan Gambar 4.15 maka akan dijelaskan mengenai langkah-langkah pada inisialisi *input weight* dan bias:

1. Memasukkan nilai *rangemin*, nilai *rangemax*, jumlah *hidden neuron*, dan jumlah data
2. Inisialisasi *input weight* dan bias dilakukan secara *random* dengan interval antara -1 sampai 1
3. Didapatkan nilai *input weight* dan bias



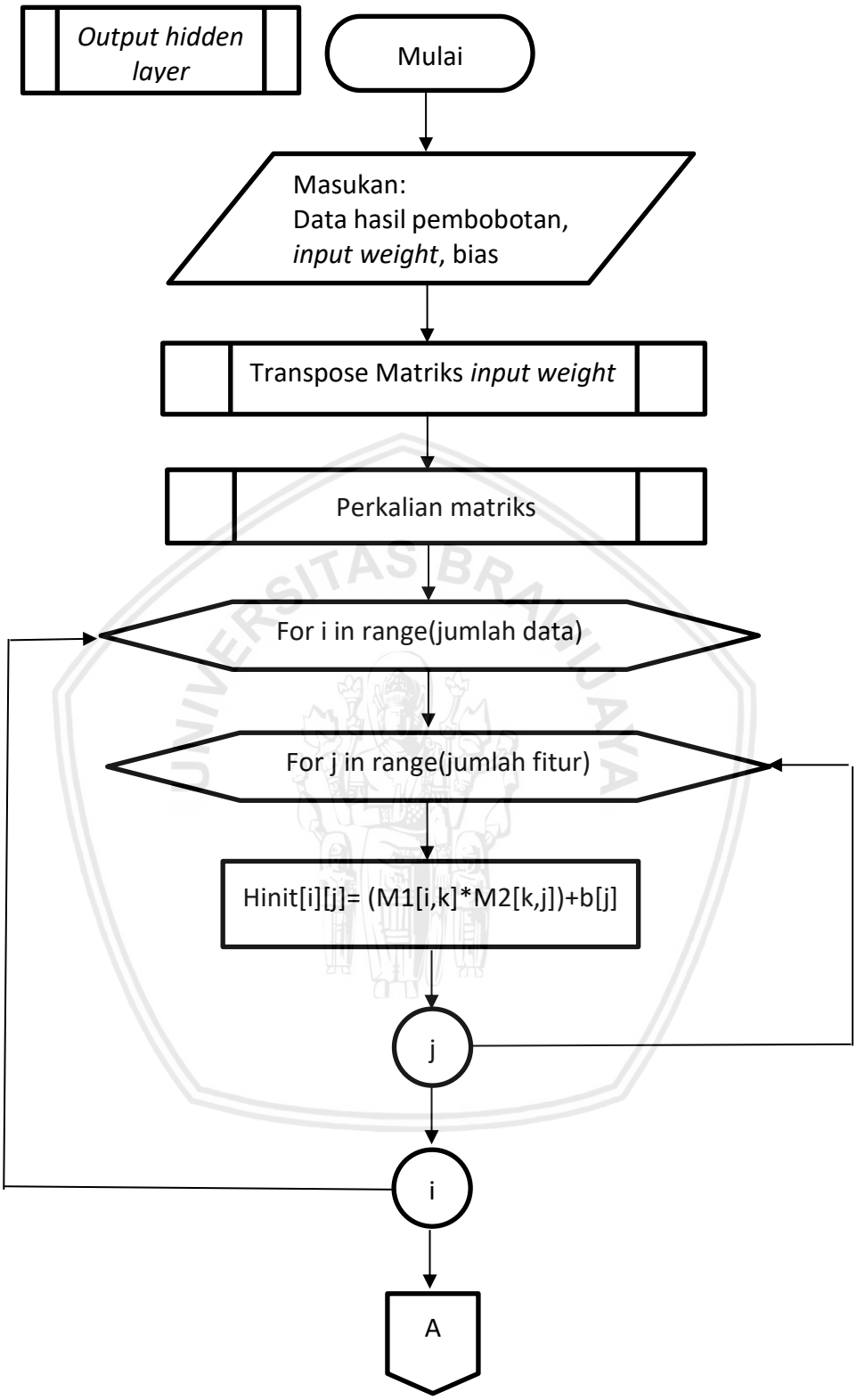
Gambar 4.15 Diagram alir inisialisasi *input weight* dan bias

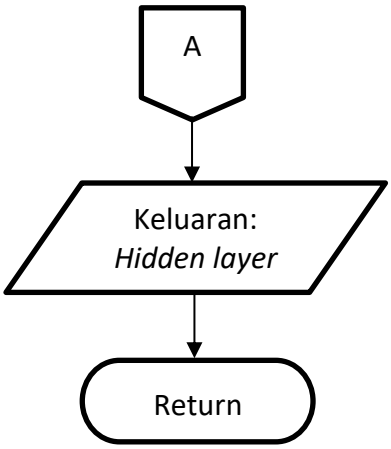
4.4.2.2 Perhitungan Output Hidden Layer

Perhitungan nilai *output hidden layer* berdasarkan Persamaan 2.5 kemudian dilakukan perhitungan dengan fungsi aktivasi berdasarkan Persamaan 2.11. Pada Gambar 4.16 akan menunjukkan detail mengenai *output hidden layer*.

Sesuai dengan perhitungan *output hidden layer* pada Gambar 4.16, berikut tahapan-tahapan diantaranya yaitu:

1. Nilai masukkan berupa data hasil pembobotan, nilai *input weight* dan bias secara *random*
2. Melakukan transpose matriks nilai *input weight*
3. Menghitung perkalian matriks data tersebut dengan matriks *input weight* yang telah di transpose
4. Menghitung *output hidden layer* berdasarkan Persamaan 2.5
5. Mendapatkan nilai *output hidden layer*.


Output hidden layer
Mulai
Masukan:
Data hasil pembobotan,
input weight, bias
Transpose Matriks input weight
Perkalian matriks
For i in range(jumlah data)
For j in range(jumlah fitur)
Hinit[i][j]= (M1[i,k]*M2[k,j])+b[j]
j
i
A



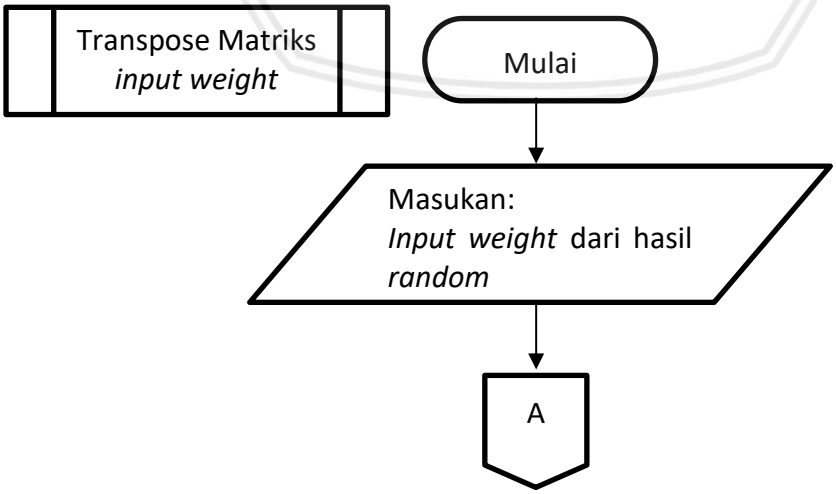
Gambar 4.16 Diagram alir *output hidden layer*

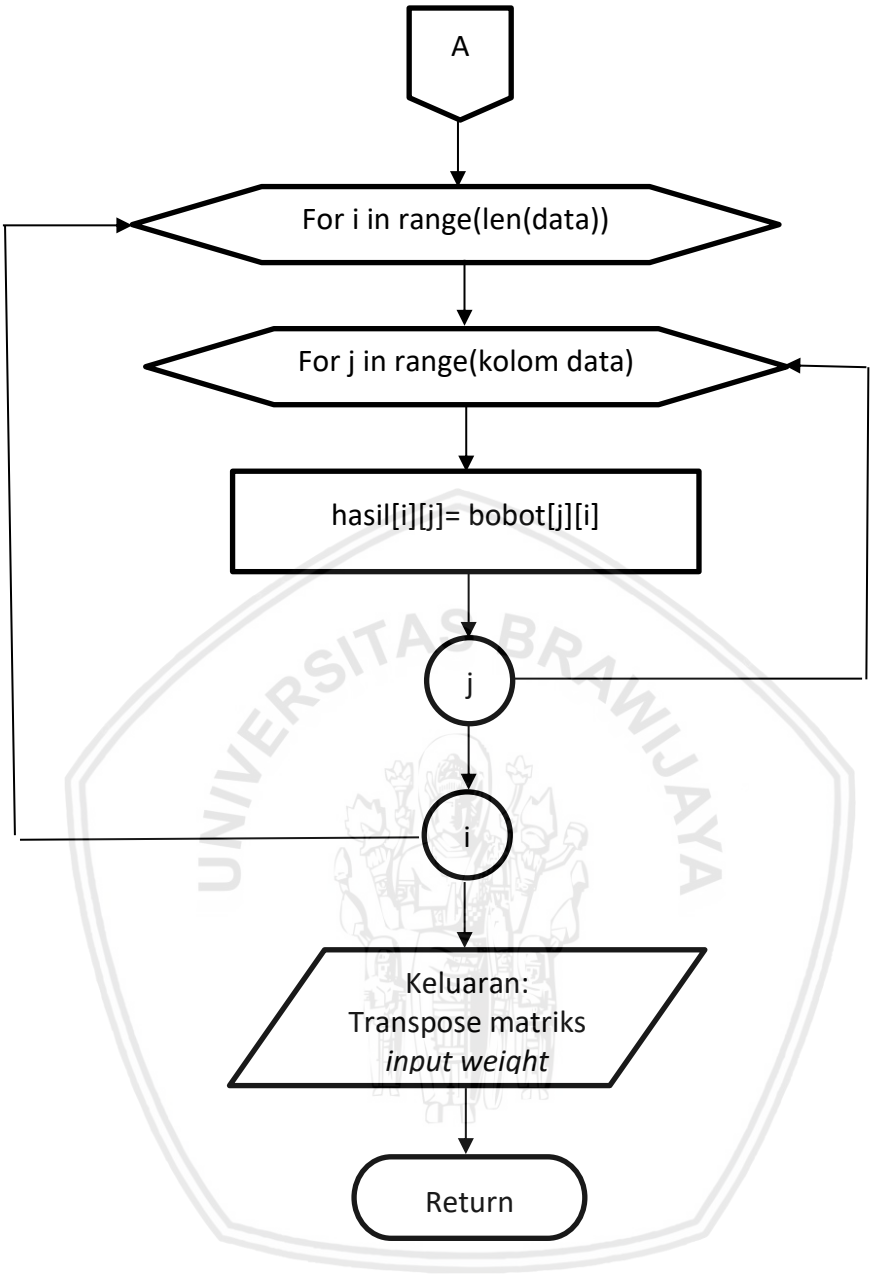
4.4.2.3 Transpose Matriks Input Weight

Dengan mendapatkan nilai *input weight* secara *random* yang dilakukan pada awal perhitungan maka dilanjutkan dengan melakukan transpose matriks *input weight* untuk perhitungan selanjutnya. Pada Gambar 4.17 akan ditunjukkan mengenai diagram alir transpose matriks *input weight*.

Sesuai dengan tahapan-tahapan transpose matriks *input weight* pada Gambar 4.17, maka akan dijelaskan sebagai berikut:

1. Nilai masukkan berupa *input weight* dengan *range* -1 hingga 1 secara *random*
2. Perulangan dilakukan dengan merubah baris menjadi kolom dan begitupun sebaliknya
3. Didapatkan hasil berupa nilai matriks transpose *input weight*





Gambar 4.17 Diagram alir transpose matriks *input weight*

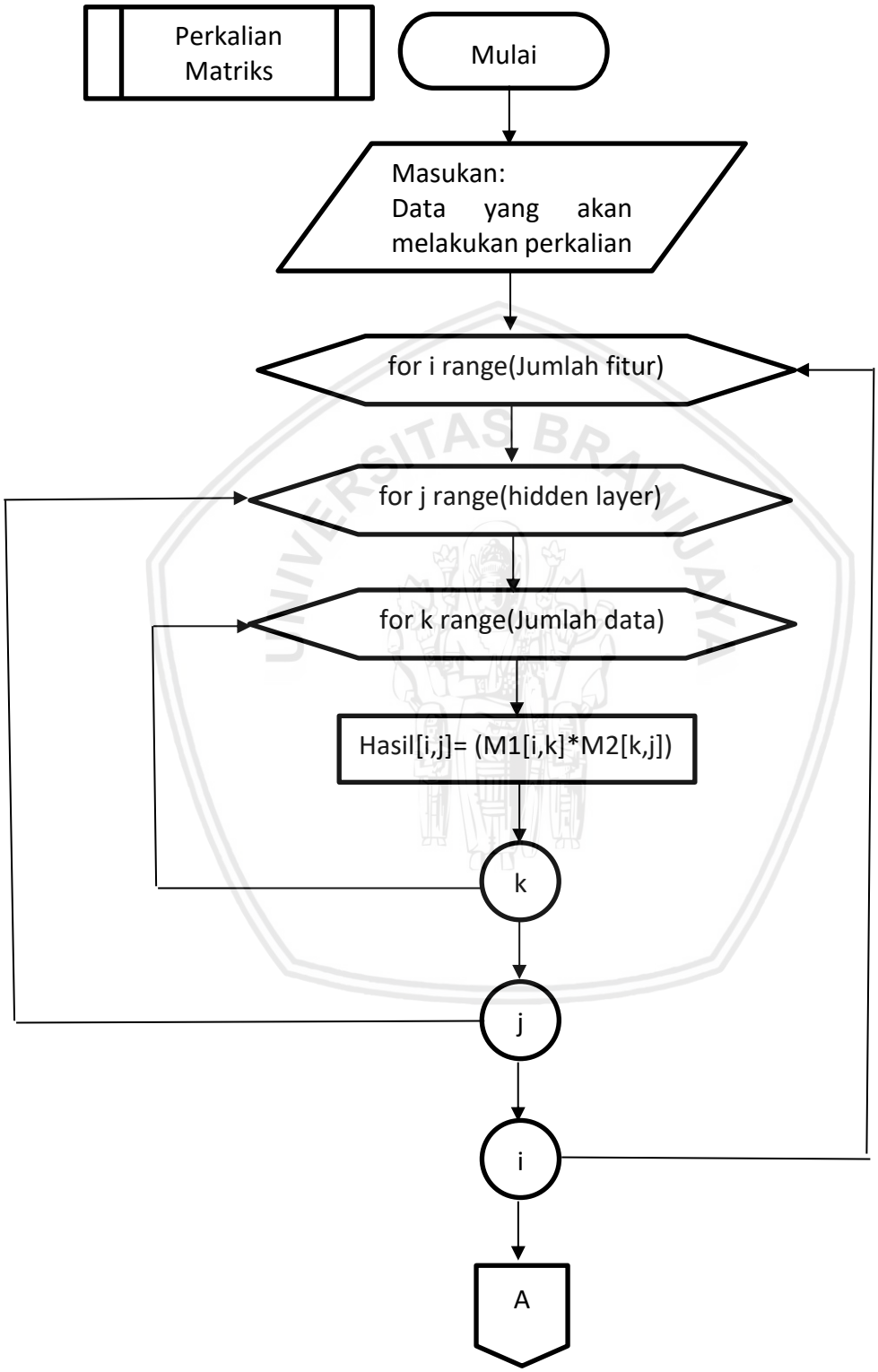
4.4.2.4 Perkalian Matriks

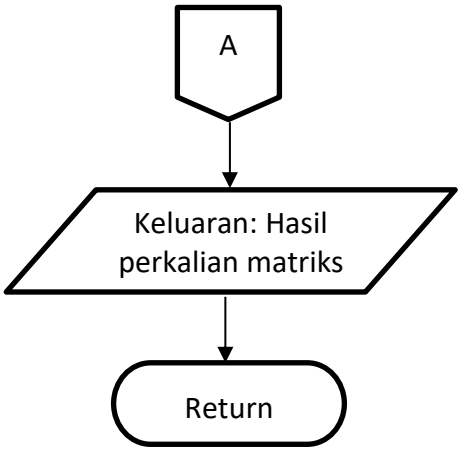
Setelah melakukan transpose *input weight* maka selanjutnya dilakukan perkalian antara data training dengan transpose *input weight* yang terdapat dalam perhitungan *output hidden layer*. Pada Gambar 4.18 akan menjelaskan mengenai diagram alir perkalian matriks.

Sesuai dengan diagram alir pada Gambar 4.18, berikut penjelasan masing-masing tahapan sebagai berikut:

1. Nilai masukkan berupa data yang akan melakukan perkalian

2. Perulangan pada baris dan kolom serta dilakukan perkalian matriks antara data perkalian
3. Mendapatkan hasil perkalian matriks

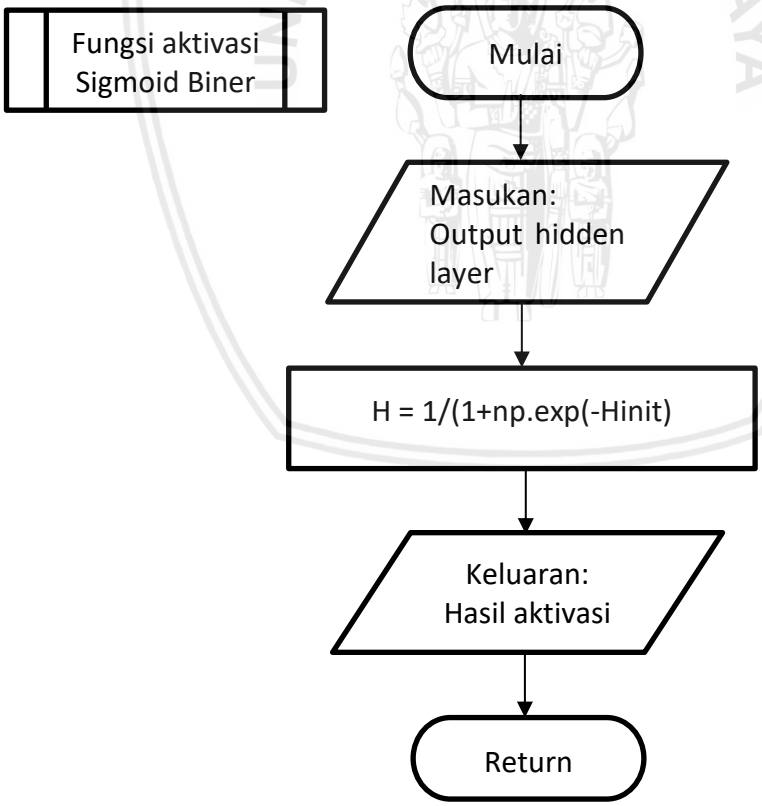




Gambar 4.18 Diagram alir perkalian matriks

4.4.2.5 Perhitungan Fungsi Aktivasi Sigmoid Biner

Proses perhitungan fungsi aktivasi Sigmoid Biner berdasarkan hasil perkalian matriks data training dengan transpose *weight*. Pada Gambar 4.19 akan dijelaskan pada diagram alir fungsi aktivasi Sigmoid Biner.



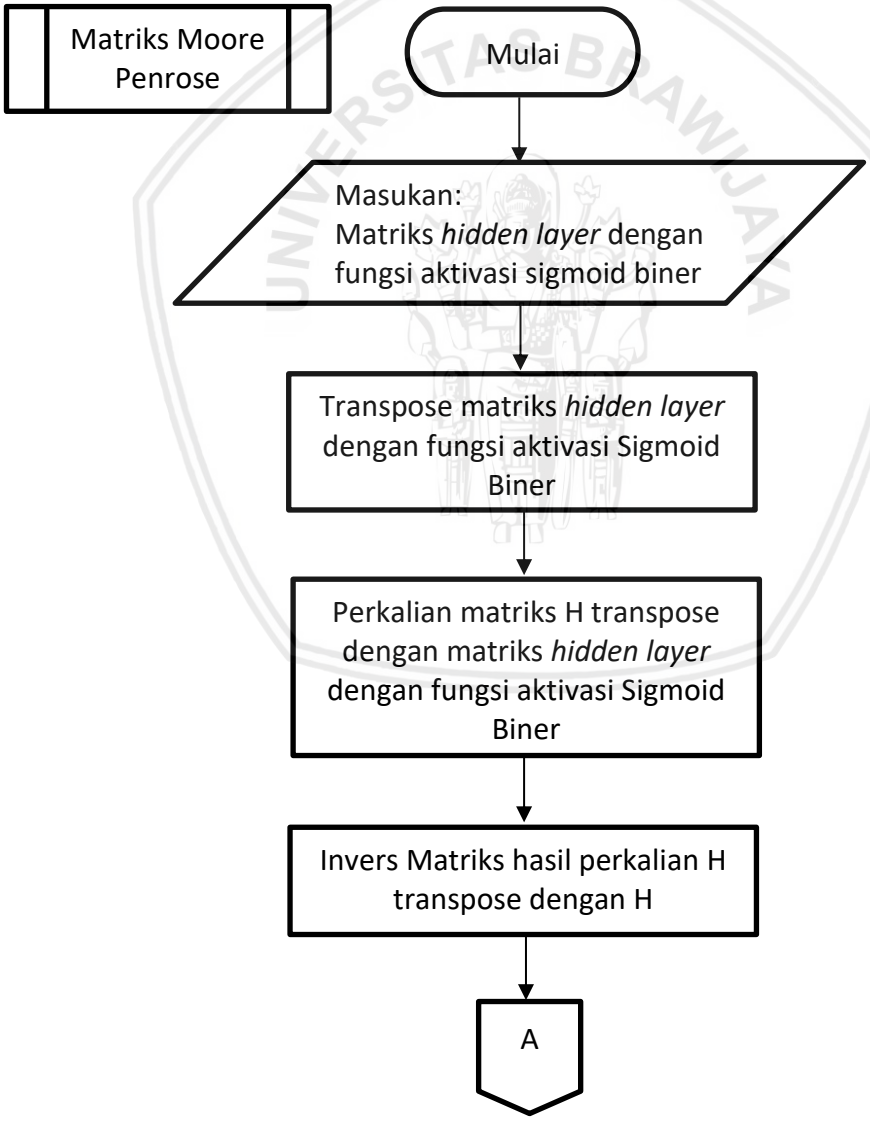
Gambar 4.19 Diagram alir fungsi aktivasi Sigmoid Biner

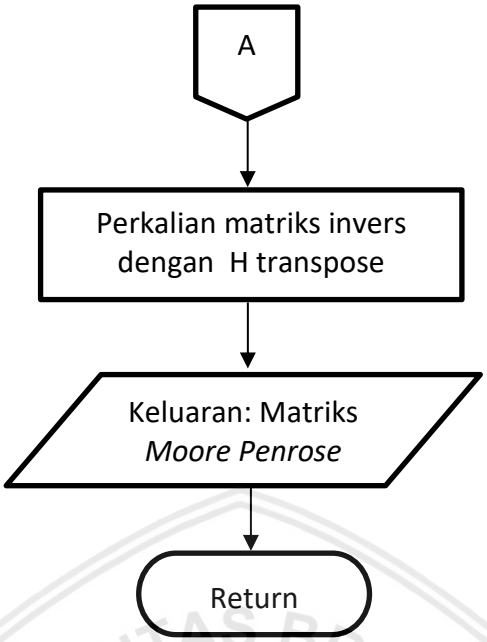
Sesuai dengan diagram alir pada Gambar 4.19, berikut penjelasan diantaranya yaitu:

1. Masukkan merupakan nilai hasil perkalian dari data *training* dengan matriks transpose *weight*
2. Selanjutnya dimasukkan ke dalam fungsi aktivasi menggunakan sigmoid biner
3. Mendapatkan nilai keluaran berupa *hidden layer* dengan fungsi aktivasi

4.4.2.6 Perhitungan Matriks Moore-Penrose

Sebelum menghitung nilai *output weight*, dilakukan perhitungan matriks *Moore-Penrose* berdasarkan Persamaan 2.6. Pada Gambar 4.20 akan dijelaskan diagram alir mengenai perhitungan *Moore-Penrose*.





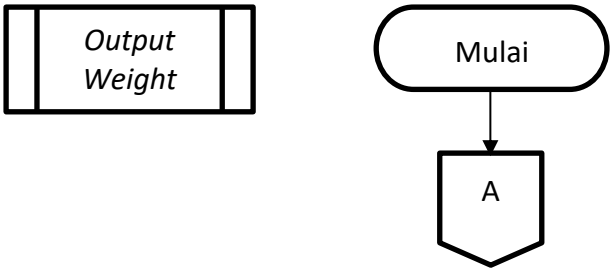
Gambar 4.20 Diagram alir matriks *Moore Penrose*

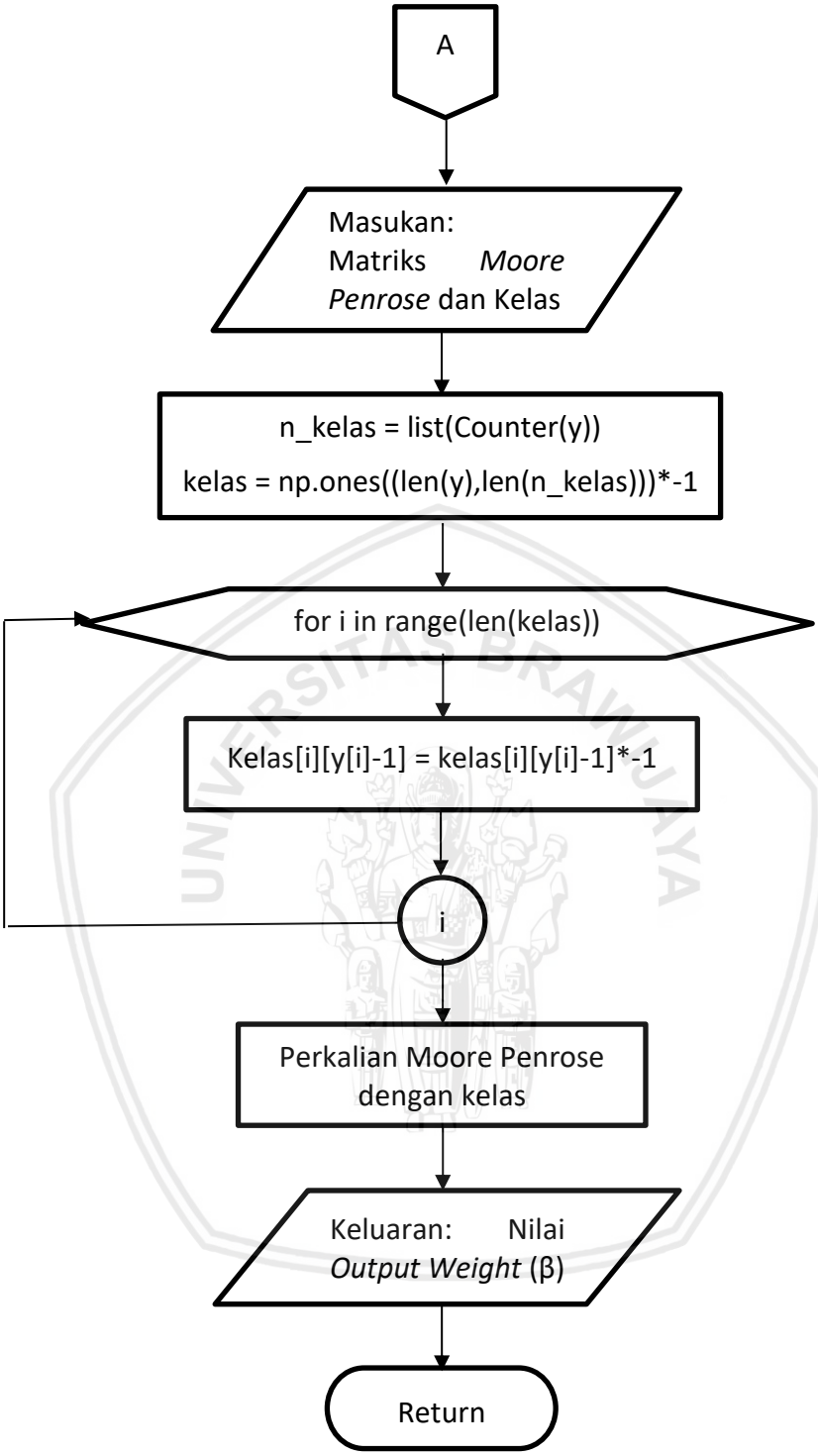
Sesuai dengan Gambar 4.20 pada diagram alir *Moore-Penrose* akan dijelaskan tahapan-tahapan sebagai berikut:

1. Nilai masukkan berupa matriks *hidden layer* dengan fungsi aktivasi Sigmoid Biner.
2. Proses Transpose matriks *hidden layer* dikalikan dengan matriks *hidden layer* dengan fungsi aktivasi Sigmoid Biner.
3. Mendapatkan hasil berdasarkan matriks sebelumnya dan setelah itu dilakukan *inverse* matriks menggunakan OBE.
4. Hasil inverse tersebut dikalikan dengan matriks transpose *hidden layer.*

4.4.2.7 Perhitungan Output Weight

Langkah akhir pada proses *training* yaitu menghitung nilai *output weight* sebelum menghasilkan nilai prediksi. Lebih detailnya akan dijelaskan pada diagram alir Gambar 4.21.





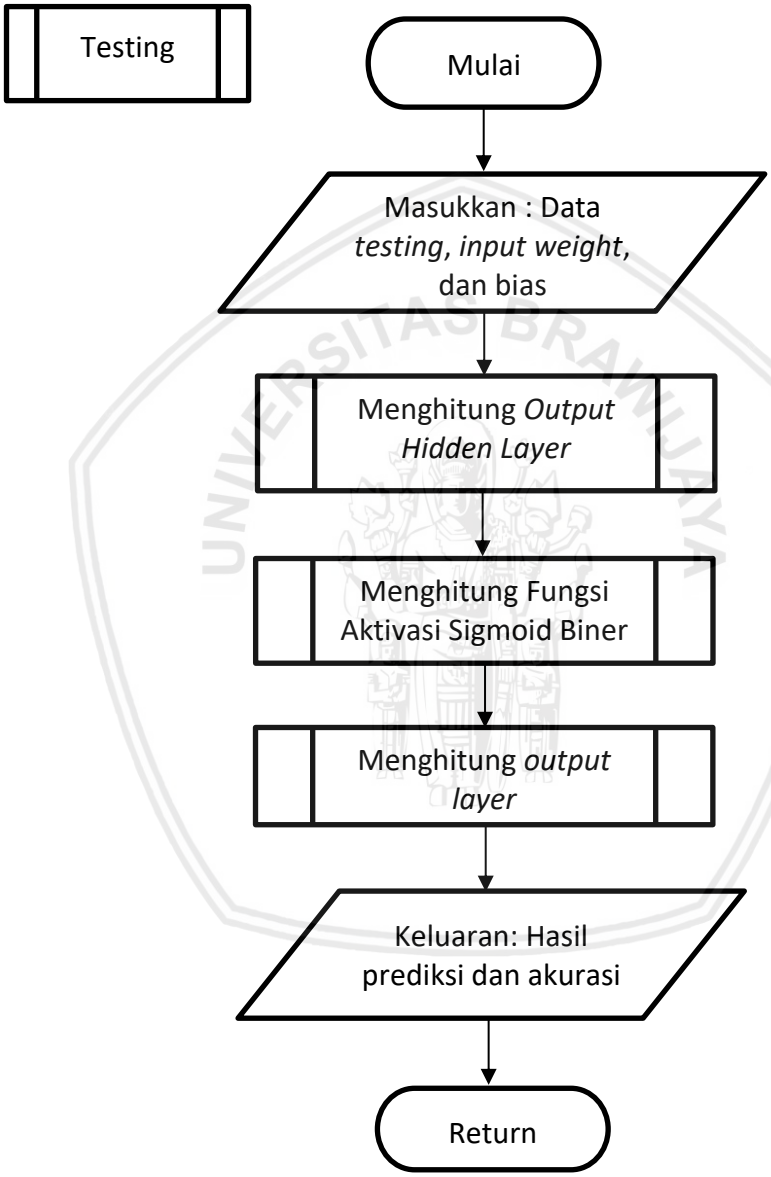
Gambar 4.21 Diagram alir *output weight*

Sesuai dengan diagram alir pada Gambar 4.21 maka penjelasan langkahnya diantara:

1. Nilai masukkan berupa matriks *Moore Penrose* dan kelas.
2. Menghitung nilai *Output Weight* dengan Persamaan 2.7.
3. Menghitung nilai *Output Weight* didapatkan dari perkalian *Moore-Penrose* dengan matriks target.

4.4.2.8 Proses Testing

Setelah tahap *training* selesai maka akan mendapatkan nilai *output weight* yang akan digunakan pada proses *testing*. Proses testing dilakukan dengan tahap yang sama dengan data *training* tetapi tanpa menghitung nilai *moore-penrose*, langsung menghitung nilai prediksi dengan akurasi. Pada Gambar 4.22 akan dijelaskan mengenai gambaran proses *testing*.



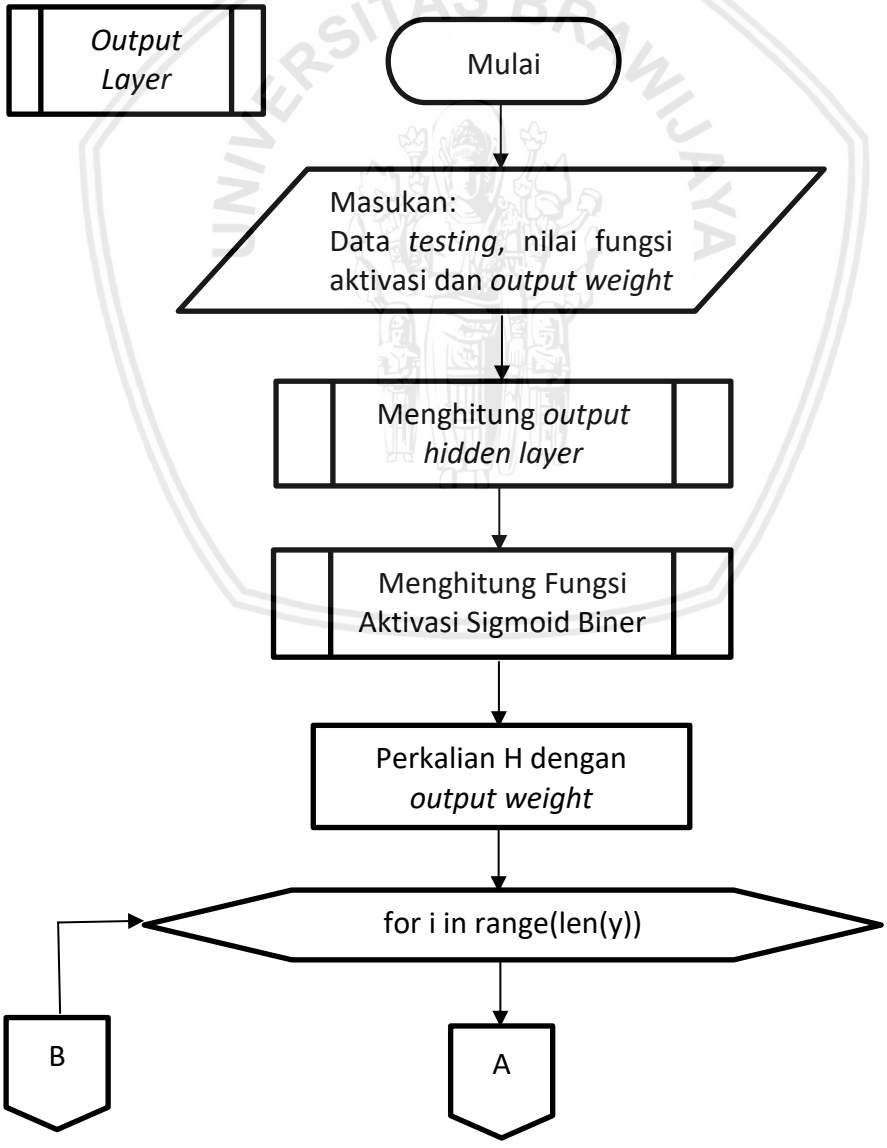
Gambar 4.22 Diagram alir data *testing*

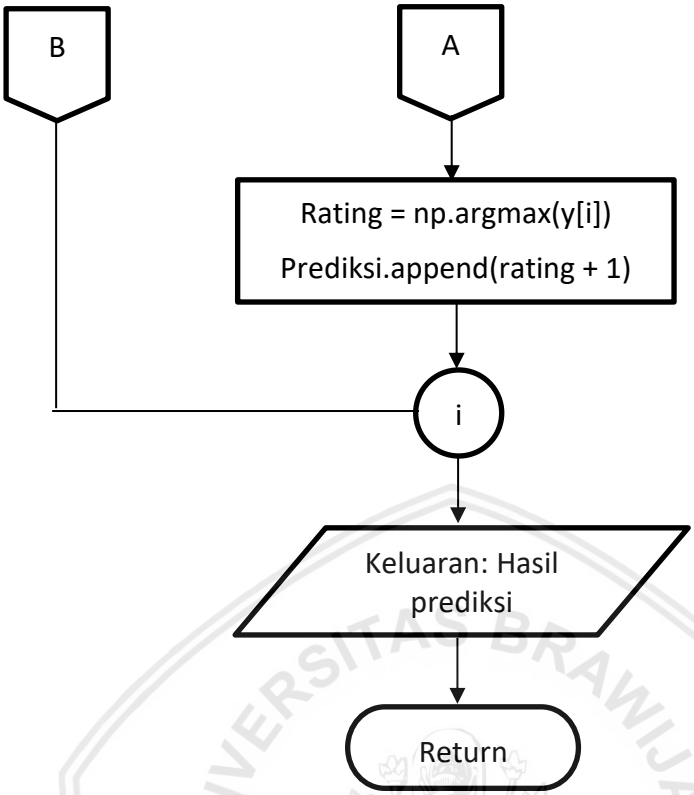
Sesuai Gambar 4.22, maka akan dijelaskan mengenai gambaran proses *testing*, yaitu:

1. Nilai masukkan berupa data *testing*, *input weight*, dan nilai bias
2. Menghitung nilai *output hidden layer*
3. Selanjutnya menghitung fungsi aktivasi menggunakan Sigmoid Biner
4. Menghitung nilai prediksi dengan *output weight* yang telah didapatkan pada proses *training*

4.4.2.9 Proses Perhitungan Output Layer

Tahap akhir yang dilakukan pada proses *testing* yaitu perhitungan *output layer* dengan hasil yang telah didapatkan pada proses *training*. Untuk lebih detailnya akan di gambarkan secara umum pada Gambar 4.23.





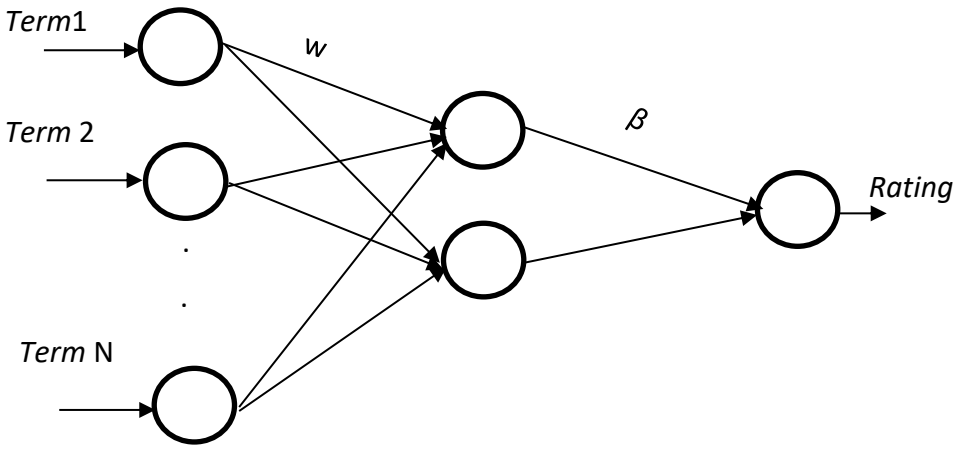
Gambar 4.23 Diagram alir *output layer*

Sesuai dengan diagram alir pada Gambar 4.23 maka penjelasan langkah dari *output layer* adalah:

1. Nilai masukkan berupa data *testing*, nilai fungsi aktivasi Sigmoid Biner, dan *output weight*
2. Perhitungan perkalian fungsi aktivasi sigmoid biner dengan matriks *output weight*
3. Mendapatkan nilai prediksi atau *output layer*

4.5 Perhitungan Manual

Pada subbab ini akan menunjukkan arsitektur *Extreme Learning Machine* pada proses manualisasi tersebut. Gambaran proses *Extreme Learning Machine* ini menunjukkan manualisasi dari proses klasifikasi *rating*. Arsitektur ini meliputi 23 *input layer*, 2 *hidden layer neuron*, dan *output layer* berupa *rating*.



23 input *layer*	2 *hidden layer neurons*	Output *layer*

Proses klasifikasi dengan 6 data latih dan 2 data uji pada Tabel 4.1 dan Tabel 4.2, proses manualisasi data *review* tersebut diawali dengan proses *pre-processing* yang terdiri dari *case folding*, *filtering*, tokenisasi, *stopword removal*, dan *stemming*.

Tabel 4.1 Data *training*

No	*Review*	*Rating*
1	Pesan ice coffee tapi pahit sekali dan barista gak ramah	2
2	pesan waffle lama banget keluarnya dan baristanya jutek sekali padahal tempatnya cozy	2
3	liat wafflenya enak tapi rasanya biasa aja	2
4	kecewa banget sama pelayanannya barista jutek sekali	1
5	pelayanan dan kopinya sangat kurang sekali!!!!!!	1
6	Saya pesan waffle dan rasanya asin sekali, pelayanannya gak bagus	1

Tabel 4.2 Data *testing*

No	Review	Rating
1	pelayanannya gak ramah sama sekali….Barista jutek	2
2	kecewa sekali dengan pelayanannya apalagi dengan barista!	1

4.5.1 *Case Folding*

Proses case folding dengan mengganti huruf kapital pada data *training* dan data *testing* menjadi *lower case*, seperti pada Tabel 4.3 dan Tabel 4.4

Tabel 4.3 *Case folding* data *training*

No	*Review*	*Rating*
1	pesan ice coffee tapi pahit sekali dan barista gak ramah	2
2	pesan waffle lama banget keluarnya dan baristanya jutek sekali padahal tempatnya cozy	2
3	liat wafflenya enak tapi rasanya biasa aja	2
4	kecewa banget sama pelayanannya barista jutek sekali	1
5	pelayanan dan kopinya sangat kurang sekali!!!!!!	1
6	saya pesan waffle dan rasanya asin sekali, pelayanannya gak bagus	1

Tabel 4.4 *Case folding* data *testing*

No	*Review*	*Rating*
1	pelayanannya gak ramah sama sekali….barista jutek	2
2	kecewa sekali dengan pelayanannya apalagi dengan barista!	1

4.5.2 *Filtering*

Proses ini dilakukan setelah melakukan proses *case folding* yaitu proses *filtering* dimana data *training* dan data *testing* akan dihilangkan angka ataupun tanda baca. Hasil filtering ditunjukkan pada Tabel 4.5 dan 4.6.

Tabel 4.5 *Filtering* data *training*

No	*Review*	*Rating*
1	pesan ice coffee tapi pahit sekali dan barista gak ramah	2
2	pesan waffle lama banget keluarnya dan baristanya jutek sekali padahal tempatnya cozy	2
3	liat wafflenya enak tapi rasanya biasa aja	2
4	kecewa banget sama pelayanannya barista jutek sekali	1
5	pelayanan dan kopinya sangat kurang sekali	1
6	saya pesan waffle dan rasanya asin sekali pelayanannya gak bagus	1

Tabel 4.6 *Filtering* data *testing*

No	*Review*	*Rating*
1	pelayanannya gak ramah sama sekali barista jutek	2
2	kecewa sekali dengan pelayanannya apalagi dengan barista	1

4.5.3 Tokenisasi

Langkah selanjutnya setelah proses *filtering* yaitu memecah kalimat menjadi *term* atau kata pada data *training* dan data *testing* atau disebut dengan tokenisasi. Hasil tokenisasi ditunjukkan pada Tabel 4.7 dan Tabel 4.8

Tabel 4.7 Tokenisasi data *training*

No	*Review*	*Rating*
1	[‘pesan’, ‘ice’, ‘coffee’, ‘tapi’, ‘pahit’, ‘sekali’, ‘dan’, ‘barista’, ‘gak’, ‘ramah’]	2
2	[‘pesan’, ‘waffle’, ‘lama’, ‘banget’, ‘keluarnya’, ‘dan’, ‘baristanya’, ‘jutek’, ‘sekali’, ‘padahal’, ‘tempatnya’, ‘cozy’]	2
3	[‘liat’, ‘wafflenya’, ‘enak’, ‘tapi’, ‘rasanya’, ‘biasa’, ‘aja’]	2
4	[‘kecewa’, ‘banget’, ‘sama’, ‘pelayanannya’, ‘barista’, ‘jutek’, ‘sekali’]	1
5	[‘pelayanan’, ‘dan’, ‘kopinya’, ‘sangat’, ‘kurang’, ‘sekali’]	1
6	[‘saya’, ‘pesan’, ‘waffle’, ‘dan’, ’rasanya’, ‘asin’, ‘sekali’, ‘pelayanannya’, ‘gak’, ‘bagus’]	1

Tabel 4.8 Tokenisasi data *testing*

No	*Review*	*Rating*
1	[‘pelayanannya’, ‘gak’, ‘ramah’, ‘sama’, ‘sekali’, ‘barista’, ‘jutek’]	2
2	[‘kecewa’, ‘sekali’, ‘dengan’, ‘pelayanannya’, ‘apalagi’, ‘dengan’, ‘barista’]	1

4.5.4 *Stopword Removal*

Proses *stopword removal* merupakan hasil dari tokenisasi, dengan menghilangkan kata-kata tidak penting yang terdapat dalam daftar *stopword*. Hasil *stopword removal* ditunjukkan pada Tabel 4.9 dan Tabel 4.10

Tabel 4.9 *Stopword removal* data *training*

No	*Review*	*Rating*
1	[‘pesan’, ‘ice’, ‘coffee’, ‘pahit’, ‘barista’, ‘gak’, ‘ramah’]	2
2	[‘pesan’, ‘waffle’, ‘banget’, ‘keluarnya’, ‘baristanya’, ‘jutek’, ‘tempatnya’, ‘cozy’]	2
3	[‘liat’, ‘wafflenya’, ‘enak’, ‘rasanya’, ‘aja’]	2
4	[‘kecewa’, ‘banget’, ‘pelayanannya’, ‘barista’, ‘jutek’]	1
5	[‘pelayanan’, ‘kopinya’, ‘kurang’]	1
6	[‘pesan’, ‘waffle’, ’rasanya’, ‘asin’, ‘pelayanannya’, ‘gak’, ‘bagus’]	1

Tabel 4.10 *Stopword removal* data *testing*

No	*Review*	*Rating*
1	[‘pelayanannya’, ‘gak’, ‘ramah’, ‘barista’, ‘jutek’]	2
2	[‘kecewa’, ‘pelayanannya’, ‘barista’]	1

4.5.5 *Stemming*

Pada proses *stemming* setelah proses *stopword removal* pada data *training* dengan data *testing* merupakan merubah semua kata menjadi kata dasar. Berikut adalah hasil stemming pada Tabel 4.11 dan Tabel 4.12

Tabel 4.11 *Stemming* data *training*

No	*Review*	*Rating*
1	[‘pesan’, ‘ice’, ‘coffee’, ‘pahit’, ‘barista’, ‘gak’, ‘ramah’]	2
2	[‘pesan’, ‘waffle’, ‘banget’, ‘luar’, ‘barista’, ‘jutek’, ‘tempat’, ‘cozy’]	2
3	[‘liat’, ‘waffle’, ‘enak’, ‘rasa’, ‘aja’]	2
4	[‘kecewa’, ‘banget’, ‘layan’, ‘barista’, ‘jutek’]	1
5	[‘layan’, ‘kopi’, ‘kurang’]	1
6	[‘pesan’, ‘waffle’, ’rasa’, ‘asin’, ‘layan’, ‘gak’, ‘bagus’]	1

Tabel 4.12 *Stemming* data *testing*

No	*Review*	*Rating*
1	[‘layan’, ‘gak’, ‘ramah’, ‘barista’, ‘jutek’]	2
2	[‘kecewa’, ‘layan’, ‘barista’]	1

4.5.6 Hasil TF-IDF

Setelah melakukan *pre-processing* maka selanjutnya akan melakukan proses pembobotan kata dengan menggunakan TF-IDF. Hasil perhitungan TF pada Tabel 4.13.

$$tf_{1,1} = 1 + log10\ (n)$$
$$tf_{1,1} = 1 + log10\ (1)$$
$$= 1$$

Tabel 4.13 Hasil perhitungan TF

	pesan	ice	coffee	pahit	…	kopi	kurang	asin	bagus
L1	1	1	1	1	…	0	0	0	0
L2	1	0	0	0	…	0	0	0	0
L3	0	0	0	0	…	0	0	0	0
L4	0	0	0	0	…	0	0	0	0
L5	0	0	0	0	…	1	1	0	0
L6	1	0	0	0	…	0	0	1	1
U1	0	0	0	0	…	0	0	0	0
U2	0	0	0	0	…	0	0	0	0

Keterangan:

L1 – L6 = data latih

U1 – U2 = data uji

Setelah mendapatkan hasil perhitungan TF maka langsung menghitung nilai DF_t. Untuk menghitung nilai *idf* seperti pada berikut.

$$idf_{t\ 10,1} = log10\frac{N}{dft}$$
$$= log10\frac{6}{3}$$
$$= 0{,}30$$

Tabel 4.14 Hasil perhitungan IDF

	pesan	ice	coffee	pahit	…	kopi	kurang	asin	bagus
L1	1	1	1	1	…	0	0	0	0
L2	1	0	0	0	…	0	0	0	0
L3	0	0	0	0	…	0	0	0	0

	pesan	ice	coffee	pahit	...	kopi	kurang	asin	bagus
L4	0	0	0	0	...	0	0	0	0
L5	0	0	0	0	...	1	1	0	0
L6	1	0	0	0	...	0	0	1	1
U1	0	0	0	0	...	0	0	0	0
U2	0	0	0	0	...	0	0	0	0
DFt	3	1	1	1	...	1	1	1	1
IDFt	0,30	0,78	0,78	0,78	...	0,78	0,78	0,78	0,78

Setelah mendapatkan hasil perhitungan IDF maka selanjutnya menghitung TF-IDF. Hasil perhitungan TF-IDF maka akan ditunjukkan pada Tabel 4.15.

$$W_{1,1} = tf\ x\ idf_t$$
$$= 1\ x\ 0{,}30$$
$$= 0{,}30$$

Tabel 4.15 Hasil perhitungan TF-IDF

	pesan	ice	coffee	pahit	...	kopi	kurang	asin	bagus
L1	0,30	0,78	0,78	0,78	...	0	0	0	0
L2	0.30	0	0	0	...	0	0	0	0
L3	0	0	0	0	...	0	0	0	0
L4	0	0	0	0	...	0	0	0	0
L5	0	0	0	0	...	0,78	0,78	0	0
L6	0,30	0	0	0	...	0	0	0,78	0,78
U1	0	0	0	0	...	0	0	0	0
U2	0	0	0	0	...	0	0	0	0

4.5.7 Normalisasi

Sebelum masuk metode *Extreme Learning Machine* maka dilakukan normalisasi setelah dari pembobotan TF-IDF. Proses normalisasi pada Tabel 4.16.

Tabel 4.16 Normalisasi

	pesan	ice	coffee	pahit	...	kopi	kurang	asin	bagus
L1	0,30	0,78	0,78	0,78	...	0	0	0	0
L2	0,30	0	0	0	...	0	0	0	0
L3	0	0	0	0	...	0	0	0	0
L4	0	0	0	0	...	0	0	0	0
L5	0	0	0	0	...	0,78	0,78	0	0
L6	0,30	0	0	0	...	0	0	0,78	0,78
U1	0	0	0	0	...	0	0	0	0
U2	0	0	0	0	...	0	0	0	0
Max	0,30	0,78	0,78	0,78	...	0,78	0,78	0,78	0,78
Min	0	0	0	0	...	0	0	0	0

Setelah mencari nilai maksimum dan minimun, maka selanjutnya menghitung rumus normalisasi pada Tabel 4.17

$$d'_{1,1} = \frac{d - \min(p)}{\max(p) - \min(p)}$$

$$= \frac{0{,}30 - 0}{0{,}30 - 0}$$

$$= 1$$

Tabel 4.17 Hasil normalisasi

	pesan	ice	coffee	pahit	...	kopi	kurang	asin	bagus
L1	1	1	1	1	...	0	0	0	0
L2	1	0	0	0	...	0	0	0	0
L3	0	0	0	0	...	0	0	0	0
L4	0	0	0	0	...	0	0	0	0
L5	0	0	0	0	...	1	1	0	0
L6	1	0	0	0	...	0	0	1	1
U1	0	0	0	0	...	0	0	0	0
U2	0	0	0	0	...	0	0	0	0

4.5.8 Proses *Training*

4.5.8.1 *Inisialisasi Input Weight dan Bias*

Proses manualisasi menggunakan metode *Extreme Learning Machine* menggunakan 23 fitur dan 2 *hidden neuron*. Dengan inisialisasi *input weight* dan bias secara *random* dengan interval -1 hingga 1. Inisialisasi akan ditunjukkan pada Tabel 4.18.

Tabel 4.18 Hasil *random input weight*

W	1	2	3	4	...	20	21	22	23
1	0,68	-0,97	-0,45	0,60	⋯	-0,30	-0,18	-0,99	0,65
2	0,25	-0,67	0,02	0,73	⋯	-0,64	-0,57	-0,57	0,04

Setelah melakukan inisialisasi *input weight*, selanjutnya akan dilakukan proses transpose matriks. Pada Tabel 4.19 menunjukkan matriks transpose *input weight*.

Tabel 4.19 Hasil transpose *input weight*

W	1	2
1	0,68	0,25
2	-0,97	-0,67
3	-0,45	0,02

W	1	2
4	0,60	0,73
..	…	…
20	-0,30	-0,64
21	-0,18	-0,57
22	-0,99	-0,57
23	0,57	0,04

Dalam menghasilkan nilai *input weight* dan nilai transpose, *hidden neuron* yang digunakan sebanyak 2. Nilai bias juga didapatkan secara *random* dengan ditentukannya nilai *hidden neuron*.

Tabel 4.20 Nilai bias

b	1	2
1	0.07	0.95

4.5.8.2 Menghitung Output Hidden Layer

Pada proses menghitung nilai *output hidden layer* menggunakan Persamaan 2.5. Berikut contoh perhitungannya yaitu sebagai berikut:

$$Hinit_{1,1} = X\ training\ .W^{\mathrm{T}} + b$$
$$= \left((1\ x\ 0{,}68\ ) + \left(1\ x(-0{,}97)\right) + \cdots + (0\ x\ 0{,}04)\right) + 0{,}07$$
$$Hinit_{1,1} = -1{,}74$$

Hasil perhitungan manual yang didapatkan untuk hasil *ouput hidden layer* pada Tabel 4.21

Tabel 4.21 Hasil dari *output hidden layer*

$Hinit$	
-1,74	2,45
2,30	-0,10
2,19	1,38
2,13	-1,21
-0,86	0,32
1,65	2,51

4.5.8.3 Menghitung Aktivasi Sigmoid Biner

Selanjutnya untuk menghitung fungsi aktivasi Sigmoid Biner pada Persamaan 2.11. berikut contoh perhitungannya:.

$$H_{1,1} = \frac{1}{1 + exp(-Hinit)} = \frac{1}{1 + exp(-(-1{,}74))} = 0{,}15$$

Hasil dari aktivasi Sigmoid Biner akan ditunjukkan pada Tabel 4.22

Tabel 4.22 Hasil dari aktivasi sigmoid biner

H	
0,15	0,92
0,91	0,47
0,90	0,80
0,90	0,23
0,30	0,58
0,84	0,92

Sebelum masuk perhitungan *Moore Penrose* maka hasil dari aktivasi Sigmoid Biner akan dilakukan transpose yang akan ditunjukkan pada Tabel 4.23.

Tabel 4.23 Hasil dari transpose aktivasi sigmoid biner

H^T	0,15	0,91	...	0,84
	0,92	0,47	...	0,92

4.5.8.4 Menghitung *Moore Penrose*

Setelah mendapatkan nilai H^T maka selanjutnya menghitung nilai *Moore Penrose*. Hasil dari perhitungan *Moore Penrose* akan dicontohkan pada persamaan 2.6 dan Tabel 4.24

$$H^{+}{}_{1,1} = (H^{\mathrm{T}} .\ H)^{-1}.H^{\mathrm{T}}$$
$$= (0{,}15 + 0{,}15) + (0{,}91 + 0{,}91) + \cdots + (0{,}13 + 0{,}13)$$
$$= 0{,}15$$

Pada Tabel 4.24 akan ditunjukkan hasil dari perkalian matriks transpose *hidden layer* dengan *output hidden* layer dengan fungsi aktivasi.

Tabel 4.24 Hasil perhitungan perkalian matriks transpose *hidden layer* dengan *output hidden layer*

$(H^T . H)$	3,24	2,44
	2,44	2,96

Setelah mengitung perkalian matriks, maka selanjutnya masuk kedalam perhitungan invers menggunakan Operasi Baris Elementer atau biasa disebut dengan OBE. Dalam perhitungan invers menggunakan OBE dapat dijabarkan sebagai berikut:

1. Tahap pertama membuat matriks identitas (I). R1 merupakan baris 1 dan R2 merupakan baris 2

$$[\,H \mid I\,] = \begin{bmatrix} 3{,}24 & 2{,}44 \\ 2{,}44 & 2{,}96 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. R1 = R1 / 3,24

$$[\,H \mid I\,] = \begin{bmatrix} 1 & 0{,}75 \\ 2{,}44 & 2{,}96 \end{bmatrix} \begin{bmatrix} 0{,}30 & 0 \\ 0 & 1 \end{bmatrix}$$

3. R2 = R2 – 2,44 x R1

$$[\,H \mid I\,] = \begin{bmatrix} 1 & 0{,}75 \\ 0 & 1{,}13 \end{bmatrix} \begin{bmatrix} 0{,}30 & 0 \\ -0{,}73 & 1 \end{bmatrix}$$

4. R2 = R2 / 1,13

$$[\,H \mid I\,] = \begin{bmatrix} 1 & 0{,}75 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0{,}30 & 0 \\ -0{,}66 & 0{,}89 \end{bmatrix}$$

5. R1 = R1 – 0,75 x R2

$$[\,H \mid I\,] = \begin{bmatrix} 1 & 0, \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0{,}81 & -0{,}66 \\ -0{,}66 & 0{,}89 \end{bmatrix}$$

Tabel 4.25 Hasil perhitungan *invers*

$(H^T . H)^{-1}$	0,81	-0,66
	-0,66	0,89

Setelah mendapatkan hasil *invers* matriks maka langkah selanjutnya, dilanjutkan perhitungan matriks *moore penrose* (H^+) pada Persamaan 2.6. Pada Tabel 4.26 ditunjukkan perhitungan nilai (H^+) yang telah dilakukan pada perhitungan sebelumnya.

$$H^+_{1,1} = (H^T . H)^{-1} . H^T$$
$$= 0{,}81 . 0{,}15$$
$$= 0{,}49$$

Tabel 4.26 Hasil dari perhitungan *Moore Penrose*

H^+	-0,49	0,41	…	0,66
	0,72	-0,18	…	0,26

4.5.8.5 Menghitung Output Weight

Langkah terakhir pada proses *training* yaitu menghitung *output weight* dengan mengalikan nilai *Moore penrose* dengan hasil prediksi dengan menggunakan Persamaan 2.7.

$$\beta_{1,1} = H^{+}\ x\ \ Y$$
$$= (-0{,}49\ x\ 1) + (0{,}41\ x\ 1) + \cdots + \quad (0{,}26\ x\ (-1)$$
$$= 0{,}36$$

Hasil perhitungan *output weight* ditunjukkan pada Tabel 4.27

Tabel 4.27 Hasil dari *output weight*

β	0,36	-0,36
	-0,45	0,45

4.5.9 Proses *Testing*

4.5.9.1 Menghitung *Output Weight*

Setelah proses tahap *training* selesai maka akan mendapatnya nilai *input weight*, bias, dan *output weight* yang akan digunakan pada proses *testing*. Berikut adalah tahapan yang terdapat pada proses *testing.*

4.5.9.2 Menghitung *Output hidden Layer*

Setelah proses *training* telah selesai maka dilanjutkan dengan menghitung nilai *output hidden layer* menggunakan bobot dan bias pada proses *training*. Berikut perhitungan *output hidden layer* pada proses *testing*.

$$Hinit_{1,1} = X\ testing * W^{\mathrm{T}} + b$$
$$= \Big((0x\ 0{,}68) + \big(0\ x(-0{,}97)\big) + \cdots + (0\ x\ 0{,}04)\Big) + 0{,}07$$
$$Hinit\ \ = -0{,}67$$

Hasil *output hidden layer* pada proses *testing* pada Tabel 4.28.

Tabel 4.28 Hasil output *hidden layer* pada data *testing*

H_{init}	
-0,67	0,78
0,92	-0,20

4.5.9.3 Menghitung fungsi aktivasi Sigmoid Biner

Setelah menghitung H_{init} maka akan mendapatkan hasil *output hidden layer* maka selanjutnya masuk kedalam perhitungan fungsi aktivasi sigmoid biner berdasarkan Persamaan 2.11.

$$H_{1,1} = \frac{1}{1 + exp(-Hinit)} = \quad \frac{1}{1 + exp(-(-0{,}67))} = 0{,}33$$

Hasil dari fungsi aktivasi Sigmoid Biner ditunjukkan pada Tabel 4.29

Tabel 4.29 Hasil dari fungsi aktivasi Sigmoid Biner pada data *testing*

H	
0,33	0,68
0,71	0,44

4.5.9.4 Menghitung ŷ prediksi

Perhitungan terakhir yaitu menghitung ŷ prediksi menggunakan Persamaan 2.9. Setelah mendapat nilai ŷ prediksi maka dicari nilai maksimum untuk merepresentasikan kelas pada data baru tersebut.

$$\hat{y}_{1,1} = H \,.\, \beta$$
$$= (0{,}33 \,.\, 0{,}35) + (0{,}68 \,.\, (0{,}4))$$
$$= -0{,}19$$

Hasil dari nilai ŷ ditunjukkan pada Tabel 4.30

Tabel 4.30 Hasil nilai ŷ

H	
-0,19	0,19
0,05	-0,05

Setelah mendapatkan nilai ŷ maka selanjutnya untuk menghitung nilai prediksi. Pada Tabel 4.31 akan ditunjukkan perhitungan hasil prediksi.

Tabel 4.31 Hasil nilai prediksi

ŷ	Rating 1	Rating 2
Uji 1	-0,19	0,19
Uji 2	0,05	-0,05

Setelah mendapatkan nilai prediksi maka akan mencari nilai maksimum pada *rating* 1 dan *rating* 2 untuk merepresentasikan kelas proses klasifikasi. Hasil klasifikasi ditunjukkan pada Tabel 4.32

Tabel 4.32 Hasil klasifikasi *rating*

ŷ	*Rating* 1	*Rating* 2	Nilai Maks	Prediksi	Benar
Uji 1	-0,19	0,19	0,19	Rating 2	1
Uji 2	0,05	-0,05	0,05	Rating 1	1

4.5.9.5 Hasil Akurasi

Akurasi yang didapatkan pada proses *testing* yaitu pada Persamaan 2.10 dengan perhitungan akurasi sebagai berikut

$$Akurasi = \frac{2}{2} x\ 100\% \ = 100\%$$

4.6 Perancangan Uji Coba

Setelah melakukan manualisasi maka selanjutnya dilakukan pengujian menggunakan parameter *Extreme Learning Machine* (ELM). Adapun pengujian yang akan dilakukan pada penelitian ini diantaranya:

1. Pengujian *k-fold cross validation*
2. Pengujian jumlah *hidden neuron*
3. Pengujian fungsi aktivasi

4.6.1 Pengujian *K-fold cross validation*

Pengujian ini dilakukan untuk memvalidasi pengujian metode yang telah dibangun sebelumnya. Dalam pengujian ini dilakukan pada seluruh data, yang dilakukan sebagai data *training* secara bergantian dengan data lainnya agar proses testing menghasilkan hasil yang optimal. Penggunaan nilai k dapat dilakukan dalam percobaan ppengujian data set. Pada Tabel 4.33 akan menunjukkan pengujian *k-fold cross validation.*

Tabel 4.33 Pengujian *K-Fold Cross Validation*

Jumlah k ke-	Percobaan ke-i					Rata-rata akurasi
	1	2	3	4	5	
2						
3						
5						
6						
10						

4.6.2 Pengujian jumlah *hidden neuron*

Pengujian jumlah *hidden neuron* dilakukan setelah pengujian sebelumnya selesai dan mendapatkan hasil terbaik. Selanjutnya pengujian jumlah *hidden neuron* dilakukan untuk mengetahui pengaruh akurasi yang dihasilkan. Pada pengujian ini menggunakan 5, 10, 25, 50, 75, dan 100 *hidden neuron* dengan interval -0,5 hingga 0,5 menggunakan fungsi aktivasi sigmoid biner. Berikut Tabel 4.34 pengujian jumlah *hidden neuron* tersebut.

Tabel 4.34 Pengujian jumlah *hidden neuron*

Jumlah *Neuron*	Percobaan ke-i					Rata-Rata akurasi
	1	2	3	4	5	
5						
10						
25						
50						
75						
100						

4.6.3 Pengujian fungsi aktivasi

Pengujian terakhir yaitu pengujian fungsi aktivasi, pengujian fungsi aktivasi dilakukan untuk mengetahui pengaruh akurasi yang dihasilkan. Pengujian fungsi aktivasi menggunakan hasil terbaik dari pengujian sebelum-sebelumnya. Fungsi aktivasi yang dilakukan berjumlah 3 yaitu Sigmodi biner, Sigmoid bipolar, dan Sin.

Tabel 4.35 Pengujian fungsi aktivasi

Fungsi Aktivasi	Percobaan ke-i					Rata-Rata akurasi
	1	2	3	4	5	
Sigmoid biner						
Sigmoid Bipolar						
Sin						

BAB 5 PEMBAHASAN

5.1 Implementasi Sistem

Dengan adanya hasil perancangan yang telah dibuat pada bab sebelumnya, maka pada bab ini menjelaskan implementasi yang telah dibangun untuk memprediksi *rating* berdasarkan *review*. Dalam mengimplementasikan sistemnya digunakan bahasa pemograman *phyton* menggunakan platform *phycharm*.

5.1.1 Implementasi Proses *Pre-Processing*

Pre processing dilakukan sebelum memprediksi *rating* berdasarkan *review* restoran. Pada tahapan *pre processing* menggunakan *case folding*, *filtering*, tokenisasi, *stopword removal*, dan *stemming*.

```
1     def tokenisasi(self, dokumen):
2         hasil=[] #inisialisasi array hasil
3         for i in range(0,len(dokumen)):
4             hasiltoken= dokumen[i].split(' ')
5             hasil.append(hasiltoken)
6         return hasil
7     def casefolding(self, dokumen):
8         hasil=[]
9         for i in range (0,len(dokumen)):
10            Hasilcase = dokumen[i].lower()
11            hasil.append(Hasilcase)
12        return hasil
13    def loadstopwords(self):
14        with open
15 ('C:/Users/Hp/Documents/SKRIPSI/stopword.txt','r') as file:
16            isi=file.readlines()
17            stopword = [i.strip() for i in isi]
18        return stopword
19    def stopwordremove(self, dokumen):
20        hasil=[]
21        stopword= self.loadstopwords()
22        #print(stopword)
23        for i in range (0, len(dokumen)):
24            hasildoc=[]
25            for j in range(0, len(dokumen[i])):
26                if dokumen[i][j] not in stopword:
27                    hasildoc.append(dokumen[i][j])
28            hasil.append(hasildoc)
29        return hasil
30    def stemming(self, dokumen):
31        hasil=[]
32        factory = StemmerFactory()
33        stemmer = factory.create_stemmer()
34        for i in range (0, len(dokumen)):
35            isi_doc = ' '.join(dokumen[i])
36            hasilstem= stemmer.stem(isi_doc)
37            hasil.append(hasilstem.split(' '))
38        return hasil
39    def filtering (self, dokumen)
40        hasil=[]
41        for I in range (0, len(dokumen)):
```

```
42              hasilfilter= re.sub('[^a-z]','', dokumen[i])
43              hasil.append(hasilfilter)
44          return hasil
45      def term(self, dokumen):
46          hasil=[]
47          for i in range(0, len(dokumen)):
48              for j in range(0, len(dokumen[i])):
49                  if dokumen[i][j] not in hasil:
50                          hasil.append(dokumen[i][j])
51          return hasil
52  def prep(data, jumlahDataLatih, jumlahDataUji):
53      proses = preprocessing()
54      hasilcase = proses.casefolding(data)
55      print('hasil case')
56      print(hasilcase)
57      hasilfilter = proses.filtering(hasilcase)
58      print('hasil filtering')
59      print(hasilfilter)
60      hasiltoken = proses.tokenisasi(hasilfilter)
61      print('hasil tokenisasi')
62      print(hasiltoken)
63      hasilstopword = proses.stopwordremove(hasiltoken)
64      print('hasil remove stopword')
65      print(hasilstopword)
66      hasilstemm = proses.stemming(hasilstopword)
67      print('hasil stem')
68      print(hasilstemm)
69
70      dataLatih = hasilstemm[:jumlahDataLatih]
71      dataUji = hasilstemm[(len(data) - jumlahDataUji):]
72
73      term = proses.term(dataLatih)
74      print('hasil term')
75      print(term)
76
77      print('jumlah term : ')
78      print(len(term))
79      return term, dataLatih, dataUji
```

Kode Program 5.1 Proses *Pre-Processing*

Adapun penjelasan dari Kode Program 5.1 yaitu sebagai berikut:

1. Baris 1-6 adalah proses tokenisasi
2. Baris 7-12 adalah proses *case folding*
3. Baris 13-29 adalah proses *stopword removal*
4. Baris 30-38 adalah proses *stemming*
5. Baris 39-44 adalah proses *filtering*
6. Baris 52-79 adalah proses mencetak *pre-processing*

5.1.2 Implementasi Proses Pembobotan Kata

Pembobotan kata dilakukan setelah *pre-processing* selesai. Pada tahap pembobotan kata dapat menghitung nilai TF, IDF, dan TF-IDF.

```
def TF(self, dokumen, term):
        hasilTF=[]
        for i in range(0, len(dokumen)):
            TFdoc=[]
            doku = ' '.join(dokumen[i])
            for j in range(0, len(term)):

                TFdoc.append(doku.count(term[j]))
            hasilTF.append(TFdoc)
        TFvalue = []
        for i in range(0, len(dokumen)):
            TFval = []
            for j in range(0, len(term)):
                if (hasilTF[i][j] > 0):
                    TFval.append(1 +
math.log10(float(hasilTF[i][j])))
                else:
                    TFval.append((hasilTF[i][j]))
            TFvalue.append(TFval)
        return hasilTF, TFvalue
    def IDF(self, TF,term):
        hasilIDF=[]
        for j in range(0, len(term)):
            IDFdoc=0
            for i in range (0, len(TF)): #DFT
                if TF[i][j] > 0:
                    IDFdoc= IDFdoc+1 #IDFT
            IDF= math.log10(len(TF)/IDFdoc)
            hasilIDF.append(IDF)
        return hasilIDF
    def TFIDF(self, TF, IDF):
        TFIDF=TF
        for i in range(0, len(TF)):
            for j in range(0, len(IDF)):
                TFIDF[i][j]= TF[i][j] * IDF[j]
        return TFIDF
def TFIDF(term,dokumen):
    w = bobot()
    tf, tfvalue = w.TF(dokumen, term)

    print('tf')
    print(np.shape(tf))
    for i in range(len(dokumen)):
        for j in range(len(term)):
            print(tf[i][j], end=" ")
        print('\n')
    print('tf value')
    print(np.shape(tfvalue))
    for i in range(len(dokumen)):
        for j in range(len(term)):
            print(tfvalue[i][j], end=" ")
        print('\n')
    idf = w.IDF(tf, term)
    print('idf')
    print(np.shape(idf))
    print(idf , '\n')
    tfidf = w.TFIDF(tfvalue, idf)
    print('tfidf')
    print(np.shape(tfidf))
```

```
60      for i in range(len(dokumen)):
61          for j in range(len(term)):
62              print(tfidf[i][j], end=" ")
63          print('\n')
64      return tfidf, idf
```

Kode Program 5.2 Proses Pembobotan Kata

Adapun penjelasan dari Kode Program 5.2 yaitu sebagai berikut:

1. Baris 1-20 adalah proses perhitungan *term frequency*
2. Baris 21-30 adalah proses perhitungan *invers document frequency*
3. Baris 31-36 adalah proses perhitungan *TF-IDF*
4. Baris 37-64 adalah proses mencetak TF-IDF

5.1.3 Implementasi Inisialisasi *input weight* dan bias

Untuk mendapatkan nilai *input weight* dan bias maka dilakukan nilai *random* dengan *range* -0,5 hingga 0,5. Proses *input weight* makan ditunjukkan pada Kode Program 5.3

```
1       def setNormalization(self,data):
2           x,y = np.array(data).shape
3           self.maxvalue = []
4           self.minvalue = []
5           for j in range(y):
6               min = 0
7               max = 0
8               for i in range(x):
9                   if min>data[i][j]:
10                      min = data[i][j]
11                  if max<data[i][j]:
12                      max = data[i][j]
13              self.maxvalue.append(max)
14              self.minvalue.append(min)
15
16      def getNormalizationData(self,data):
17          x,y = np.array(data).shape
18          hasil_norm = np.zeros((x,y))
19          for i in range(x):
20              for j in range(y):
21                  if self.maxvalue[j]!=self.minvalue[j]:
22                      hasil_norm[i][j] = (data[i][j]-
23  self.minvalue[j])/ (self.maxvalue[j]-self.minvalue[j])
24                  else:
25                      hasil_norm[i][j] = (data[i][j]-
26  0)/(self.maxvalue[j]-0)
27          return hasil_norm
28
29      def setBobotBias(self,Bobot,bias):
30          self.W=Bobot
31          self.b=bias
32
33      def getBobotBias(self,rangemin=-0.5,rangemax=0.5):
34          self.W =
35  np.random.uniform(low=rangemin,high=rangemax,size=(self.j,
36  self.k))
```

```
37          self.b =
38  np.random.uniform(low=rangemin,high=rangemax,size=self.j)
```

Kode Program 5.3 Inisialisasi *Input weight* dan bias

Adapun penjelasan dari Kode Program 5.3 yaitu sebagai berikut:

1. Baris 1-3 adalah proses normalisasi Min-Max
2. Baris 4-6 adalah proses insialisasi bobot dan bias
3. Baris 7-12 adalah proses nilai *random* bobot dan bias

5.1.4 Implementasi *Output Hidden Layer*

Implementasi *output hidden layer* digunakan untuk proses *training* dan proses testing. Berikut Kode Program 5.4 akan dijabarkan secara detail.

```
1     def Hinit(self,fitur):
2           x, y = fitur.shape
3           WT = self.transpose(self.W)
4           m, n = WT.shape
5           hasil = np.zeros((x, n))
6           for i in range(0, x):
7               for j in range(0, n):
8                   for k in range(0, m):
9                       hasil[i, j] = hasil[i, j] + (fitur[i, k]
10  * WT[k][j])+self.b[j]
11          return hasil
12    def transpose(self,matriks):
13          x,y = matriks.shape
14          hasil = np.zeros((y,x))
15          for i in range(x):
16              for j in range(y):
17                  hasil[j][i]= matriks[i][j]
18          return hasil
19    def perkalianMatriks(self,M1,M2):
20          x,y = M1.shape
21          m,n = M2.shape
22          hasil = np.zeros((x,n))
23          for i in range(0,x):
24              for j in range(0,n):
25                  for k in range(0,m):
26                      hasil[i,j] = hasil[i,j]
27  +(M1[i,k]*M2[k][j])
28          return hasil
```

Kode Program 5.4 *Output hidden layer*

Adapun penjelasan dari Kode Program 5.4 yaitu sebagai berikut:

1. Baris 1-10 adalah proses menghitung *output hidden layer.*
2. Baris 11-18 adalah proses transpose *input weight*
3. Baris 19-28 adalah proses perkalian data dengan transpose *input weight*

5.1.5 Implementasi Fungsi Aktivasi

Pada proses perhitungan Fungsi Aktivasi menggunakan Sigmoid Biner, Sigmoid Bipolar, dan Sin yang dilakukan pada proses *training* dan *testing*. Pada Kode Program 5.5 akan ditunjukkan mengenai implementasi Fungsi Aktivasi.

```
1     def HitungH(self,Hinit , jenis):
2          H = np.zeros((len(Hinit), len(Hinit[0])))
3          if (jenis.lower() == "sigmoid biner"):
4              H = 1 / (1 + np.exp(-Hinit))
5          elif (jenis.lower() == "sin"):
6              H = np.sin(Hinit)
7          elif (jenis.lower() == "sigmoid bipolar"):
8              H = (1-np.exp(-Hinit)) / (1+np.exp(-Hinit))
9
10         return H
```

Kode Program 5.5 Implementasi Fungsi Aktivasi

Adapun penjelasan dari Kode Program 5.5 yaitu sebagai berikut:

1. Baris 1-2 adalah proses perhitungan fungsi aktivasi dengan nilai awal 0
2. Baris 3-4 adalah proses perhitungan fungsi aktivasi Sigmoid Biner
3. Baris 5-6 adalah proses perhitungan fungsi aktivasi Sin
4. Baris 7-8 adalah proses perhitungan fungsi aktivasi Sigmoid Bipolar

5.1.6 Implementasi *Moore Penrose*

Pada proses perhitungan *Moore Penrose* dilakukan pada proses *training*. Maka implementasi *Moore Penrose* akan ditunjukkan pada Kode Program 5.6

```
1     def HitHplus(self,H):
2          HT = self.transpose(H)
3          HH = self.perkalianMatriks(HT,H)
4          InvHH = np.linalg.pinv(HH)
5          Hplus = self.perkalianMatriks(InvHH,HT)
6          print('h+')
7          print(Hplus)
8          return Hplus
```

Kode Program 5.6 Implementasi *Moore Penrose*

Adapun penjelasan dari Kode Program 5.6 yaitu sebagai berikut:

1. Baris 1-5 adalah proses perhitungan *moore penrose*
2. Baris 6-7 adalah proses print perhitungan *moore penrose*

5.1.7 Implementasi *Output Weight*

Pada proses perhitungan *Ouput Weight* dilakukan perkalian dari hasil perhitungan *Moore Penrose* dengan matriks target. Pada Kode Program 5.7 akan ditunjukkan mengenai implementasi *Output Weight*.

```
1     def Btrop(self,Hplus,Y):
2          self.B = self.perkalianMatriks(Hplus,Y)
3          return self.B
```

Kode Program 5.7 Implementasi *Output Weight*

Adapun penjelasan dari Kode Program 5.7 yaitu sebagai berikut:

1. Baris 1 adalah fungsi *output weight* dengan parameter perhitungan *moore penrose* dan target.
2. Baris 2 adalah proses perkalian antara perhitungan *moore penroose* dengan target.
3. Baris 3 adalah proses pengembalian nilai *output weight.*

5.1.8 Implementasi Proses Perhitungan Prediksi

Pada proses perhitungan prediksi dilakukan perkalian antara nilai aktivasi pada proses *testing* dengan nilai *output weight* pada proses *training*. Pada implementasi perhitungan prediksi akan ditunjukkan pada Kode Program 5.8

```
1     def hitY(self,H,Btrop=None):
2          if Btrop is None:
3              Btrop = self.B
4          Y = self.perkalianMatriks(H,Btrop)
5          return Y
6
7      def predict(self,fitur,bobot,bias,Btrop, aktivasi):
8          fitur = np.array(fitur)
9          self.setBobotBias(bobot,bias)
10         Hinit =self.Hinit(fitur)
11         H = self.HitungH(Hinit, aktivasi)
12         y = self.hitY(H,Btrop)
13         prediksi_kelas = []
14         for i in range(len(y)):
15             rating = np.argmax(y[i])
16             prediksi_kelas.append(rating + 1)
17         return y, prediksi_kelas
```

Kode Program 5.8 Implementasi proses Prediksi

Adapun penjelasan dari Kode Program 5.8 yaitu sebagai berikut:

1. Baris 1-5 adalah proses perhitungan hasil prediksi
2. Baris 6-20 adalah fungsi dari proses perhitungan *training*
3. Baris 21-31 adalah fungsi dari proses perhitungan *testing*

5.1.9 Implementasi Evaluasi menggunakan Akurasi

Pada proses evaluasi menggunakan akurasi merupakan tahap terakhir kali. Pada Kode Program 5.9 akan ditunjukkan mengenai implementasi evaluasi menggunakan akurasi.

```
1          prep_data_uji = prep(datauji)
2          tfidf_uji = TFIDF_Uji(terms, prep_data_uji, idf)
3
4          y, prediksi_kelas = pengujian(tfidf_uji, bbt, bias,
5  btrop, jenis_aktivasi[i])
```

```
6
7           yg_benar = 0
8           for i in range(len(prediksi_kelas)):
9               if prediksi_kelas[i] == int(klsdu[i]):
10                  yg_benar = yg_benar + 1
11
12          akurasi = (yg_benar / len(klsdu)) * 100
13
14          print('prediksi')
15          print(y)
16          print(klsdu)
17          print(prediksi_kelas)
18          print('akurasi : ', akurasi)
19          hasil.append(akurasi)
```

Kode Program 5.9 Implementasi Evaluasi menggunakan Akurasi

Adapun penjelasan dari Kode Program 5.9 yaitu sebagai berikut:

1. Baris 1-5 adalah proses pengujian
2. Baris 7 adalah menginisialisasi dengan awal yaitu 0
3. Baris 8-10 adalah perulangan untuk menghitung nilai prediksi
4. Baris 12 adalah rumus dari akurasi
5. Baris 14-19 adalah proses mencetak nilai akurasi

BAB 6 PENGUJIAN DAN ANALISIS

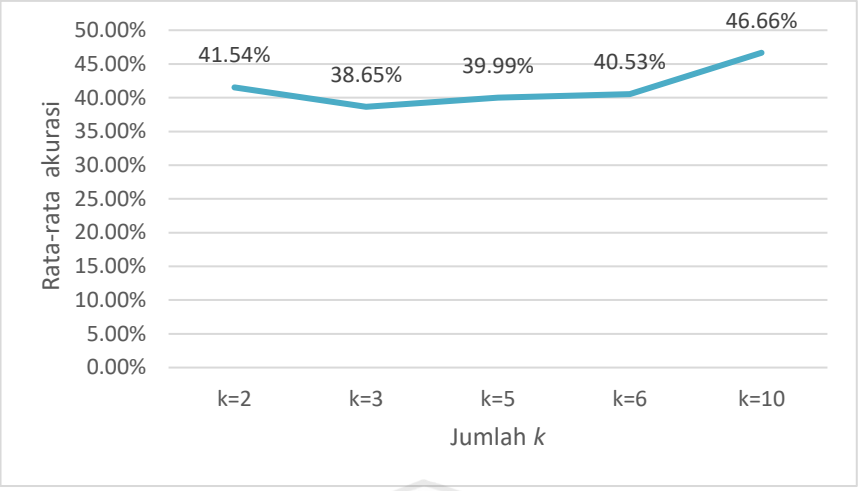
Bab ini dilakukan setelah implementasi yang dibangun telah selesai, dilakukannya pengujian untuk mengetahui performa metode yang telah diterapkan. Pada pengujian ini dilakukan 3 pengujian yaitu pengujian *k-fold cross validation*, pengujian jumlah *hidden neuron*, dan pengujian fungsi aktivasi. Pengujian jumlah *hidden neuron* dan pengujian fungsi aktivasi merupakan parameter yang terdapat pada metode ELM.

6.1 Pengujian dan Analisis *K-Fold Cross Validation*

Pengujian pertama dilakukan untuk mengetahui jumlah data *training* dan data *testing* yang ideal. Pada pengujian ini menggunakan 150 data dengan 5 kali uji coba. Pengujian ini dilakukan menggunakan *k*=2, *k*=3, *k*=5, *k*=6, dan *k*=10. Pengujian ini menggunakan *hidden neuron* sebanyak 10 dengan fungsi aktivasi sigmoid biner, interval bobot yang digunakan yaitu -0,5 hingga 0,5. Pada Tabel 6.1 akan ditunjukkan hasil pengujian *k-fold cross validation*

Tabel 6.1 Hasil Pengujian *K-Fold Cross Validation*

Jumlah nilai k	Percobaan ke-					Rata-rata
	1	2	3	4	5	
2	41,36%	42%	42%	41,36%	41%	41,54%
3	42%	41%	38,64%	38,64%	38%	38,65%
5	41,99%	**36%**	40%	38,64%	43,36%	39,99%
6	44,64%	40%	40%	38%	40%	40,53%
10	**60%**	40%	46,64%	40%	40,64%	46,66%
Rata-rata hasil akurasi						41,48%



Gambar 6.1 Grafik pengujian *k-fold*

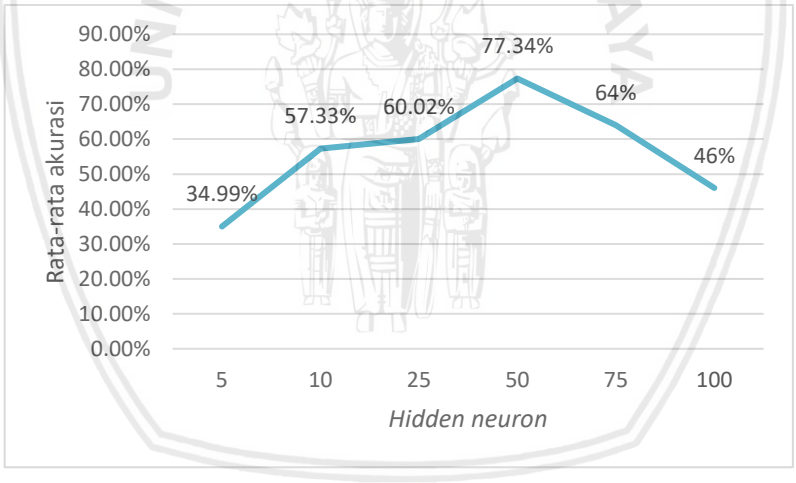
Berdasarkan Tabel 6.1 dan Gambar 6.1 menunjukkan grafik pengujian *k fold cross validation*, tujuan dari pengujian *k-fold cross validation* yaitu untuk mengetahui kesesuaian data tersebut pada metode *Extreme Learning Machine*. Dilihat dari hasil *k*=2, *k*=3, *k*=5, *k*=6, dan *k*=10 hasil tersebut tidak berbeda jauh walapun menggunakan nilai bobot dan bias secara *random.* Pada percobaan ke-2 terdapat akurasi terendah pada *k*=5 sebesar 36% dan akurasi terbesar pada percobaan ke-1 sebesar 60%. Hasil rata-rata tertinggi dari 5 percobaan pada *k*=10 yaitu 46,66% dan hasil rata-rata dari semua nilai *k* yaitu 41,48%. Selain itu hasil rata-rata terendah yaitu terdapat pada *k*=3 yaitu 38,65%. Dengan adanya hal tersebut dapat disimpulkan bahwa metode ELM, jika menggunakan nilai *k* semakin besar maka semakin banyak pula jumlah data *training* yang digunakan sehingga *term* yang dihasilkan semakin bervariasi dan hasil akurasi semakin baik.

6.2 Pengujian dan Analisis Jumlah *Hidden Neuron*

Pengujian selanjutnya yaitu jumlah *hidden neuron* tujuan dilakukan pengujian ini untuk mengetahui pengaruh parameter ELM ini terhadap akurasi. Pengujian ini menggunakan hasil terbaik pada pengujian sebelumnya yaitu dengan *k*=10 dengan penjabaran 135 data *training* dan 15 data *testing*. Pengujain ini menggunakan 5 kali uji coba dan mencari nilai akurasi paling tinggi untuk pengujian selanjutnya. Pengujian ini menggunakan jumlah *hidden neuron* sebanyak 5, 10, 25, 50, 75, dan 100. Fungsi aktivasi yang digunakan yaitu sigmoid biner dengan interval bobot dan bias yaitu -0,5 hingga 0,5. Pada Tabel 6.2 akan ditunjukkan hasil dari pengujian tersebut.

Tabel 6.2 Hasil pengujian jumlah *hidden neuron*

Jumlah Neuron	Percobaan ke-					Rata-Rata akurasi
	1	2	3	4	5	
5	**20%**	33,34%	46,64%	33,34%	46,64%	34,99%
10	46,64%	60%	80%	53,36%	46,64%	57,33%
25	80%	53,36%	60%	53,36%	53,36%	60,02%
50	73,34%	80%	**86,67%**	80%	66,57%	77,34%
75	60%	60%	60%	70%	70%	64%
100	30%	50%	40%	50%	60%	46%



Gambar 6.2 Grafik pengujian *hidden neuron*

Pada Tabel 6.2 dan Gambar 6.2 menghasilkan akurasi yang bervariasi dikarenakan nilai bobot dan bias dilakukan secara *random*, untuk hasil terendah yang dihasilkan saat jumlah *hidden neuron* sebanyak 5 dengan akurasi sebesar 20%, sedangkan akurasi tertinggi didapatkan saat jumlah *hidden* neuron sebanyak 50 saat percobaan ke-3 dengan akurasi sebesar 86,67%. Namun untuk rata-rata akurasi tertinggi dari 5 percobaan yang menghasilkan jumlah *hidden neuron* 50 yaitu sebesar 77,34%. Rata-rata akurasi terendah dari 5 percobaan terdapat pada jumlah *hidden layer* sebanyak 5 yaitu sebesar 34,99%. Sehingga dapat disimpulkan bahwa semakin besar jumlah *hidden neuron* tidak selalu menghasilkan akurasi yang baik.

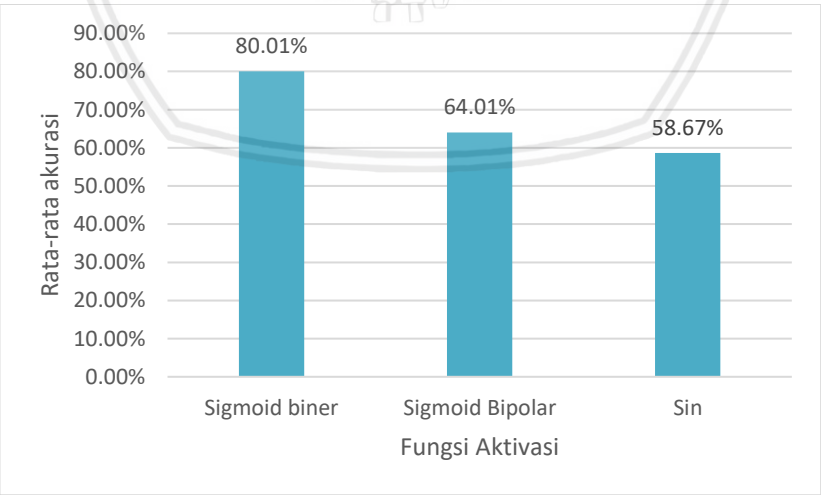
6.3 Pengujian dan Analisis Fungsi Aktivasi

Pengujian terakhir yaitu pengujian fungsi aktivasi. Pengujian fungsi aktivasi dilakukan untuk mengetahui fungsi aktivasi mana yang tepat untuk masalah prediksi *rating* berdasarkan *review* ini menggunakan metode *Extreme Learning Machine*. Penelitian ini menggunakan hasil terbaik dari pengujian-pengujian sebelumnya, diharapkan menghasilkan akurasi yang baik. Pada pengujian ini menggunakan 135 data *training* dan 15 data *testing* menggunakan *hidden neuron* sebanyak 50. Selain itu pengujian ini menggunakan interval bobot dan bias yaitu -0,5 hingga 0,5.

Proses pengujian ini menghasilkan rata-rata nilai akurasi yang berbeda-beda. Hal tersebut dikarenakan *input weight* dan bias menggunakan nilai *random*. Pada Tabel 6.3 akan menunjukkan nilai hasil pengujian fungsi aktivasi dan Gambar 6.3 akan ditunjukkan grafik hasil pengujian tersebut

Tabel 6.3 Hasil pengujian fungsi aktivasi

Fungsi Aktivasi	Subset ke-					Rata-Rata akurasi
	1	2	3	4	5	
Sigmoid biner	80%	80%	66.67%	**86.67%**	86.67%	80.01%
Sigmoid Bipolar	66.67%	**46.64%**	53.36%	73.34%	80%	64.01%
Sin	60%	53.36%	66.67%	**46.64%**	66.67%	58.67%



Gambar 6.3 Grafik pengujian fungsi aktivasi

Pada Tabel 6.3 dan Gambar 6.3 dapat dilihat bahwa masing-masing fungsi aktivasi hanya menghasilkan nilai rata-rata akurasi yang bervariasi. Maka akan terlihat dalam pengujian fungsi aktivasi yang terendah pada fungsi aktivasi Sigmoid Bipolar dan Sin sebesar 46,64%. Namun untuk akurasi tertinggi ditunjukkan pada fungsi aktivasi Sigmoid Biner sebesar 86.67%. Rata-rata akurasi dalam 5 percobaan yaitu Sigmoid Biner sebesar 80,01%. Selain itu dapat diketahui rata-rata hasil akurasi terendah sebesar 58,67%. Dapat disimpulkan bahwa masing-masing fungsi aktivasi mempunyai fungsi yang berbeda-beda. Fungsi aktivasi Sigmoid Biner merupakan fungsi aktivasi yang sesuai dengan menggunakan data biner.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdarkan hasil dari pengujian serta analisis dari pengujian tersebut mengenai prediksi *rating* otomatis berdasarkan *review* restoran pada aplikasi Zomato dengan menggunakan *Extreme Learning Machine*. Maka didapatkan kesimpulan sebagai berikut:

1. Metode *Extreme Learning Machine* (ELM) mampu menyelesaikan permasalahan prediksi *rating* otomatis berdasarkan *review* restoran pada aplikasi Zomato dengan akurasi cukup baik. Proses implementasi yang dilakukan yaitu *pre-processing* yang terdiri dari *case folding*, *filtering*, tokenisasi, *stopword removal*, dan *stemming*. Tahap selanjutnya adalah ekstraksi fitur dengan menggunakan pembobotan TF-IDF, lalu normalisasi, dan selanjutnya menggunakan *Extreme Learning Machine*.
2. Berdasarkan dari hasil pengujian prediksi *rating* otomatis berdasarkan *review* restoran pada aplikasi Zomato menggunakan *Extreme Learning Machine* dengan 150 data yang terdiri dari 135 data *training* dan 15 data *testing* saat *k*=10. Sehingga menggunakan *hidden layer* sebanyak 50 dengan fungsi aktivasi Sigmoid Biner. Pada metode *Extreme Learning Machine* (ELM) menghasilkan akurasi sebesar 80,01%. Nilai akurasi tersebut dinilai cukup baik untuk menyelesaikan permasalahan ini.

7.2 Saran

Saran yang diberikan untuk pengujian metode selanjutnya terhadap prediksi menggunakan *Extreme Learning Machine*, sebagai berikut:

1. Pada data *review* yang diperoleh dari aplikasi Zomato masih banyak dijumpai kata-kata tidak baku sehingga dapat memberikan teknik untuk mengidentifikasi fitur tersebut seperti konsep sinonim atau *thesaurus,* selain itu dapat menambahkan teknik optimasi pada data *training* agar mendapatkan data training ideal.
2. Penelitian selanjutnya untuk metode *Extreme Learning Machine* (ELM) memiliki bobot dan bias digunakan secara *random*. Sehingga diperlukan suatu teknik optimasi untuk mencari bobot dan bias terbaik agar menghasilkan akurasi yang optimal.

DAFTAR REFERENSI

Afifah, A., 2015. Peramalan Indeks Harga Saham Gabungan (IHSG) dengan Metode Extreme Learning Machine (ELM). Jurnal Matematika, Volume 43,pp.601-608.

Anicic, O., Skrijelj, H. and Nedi, B., 2017. Prediction of laser cutting heat affected zone by extreme learning machine. 88, pp.1–4.

Ganesan, K., 2015. A Brief Note on Stop Words for Text Mining and Retrieval. Available at: http://www.text-analytics101.com/2014/10/allabout-stop-words-for-text-mining.html.

Haddi, E., Liu X., dan Shi, Y. (2013). The Role of Text Pre-Processing in Sentiment Analysis. Procedia Computer Science, pp.26-23.

Huang, G. B., Zhu, Q. Y. & Siew, C. K., 2006. Extreme Learning Machine : Theory and Application. *Elsevier (Scient Direct),* Volume 70, pp. 489-501.

Jain, Y.K. & Bhandare, S. K., 2011. Min Max Normalization Based Data Pertubation Method For Prifacy Proection. Computer Science & Engineering, 2(VIII), pp. 45-50

Jatmiko, W., Mursanto, P., Fajar, M., Tawakal, M.I., Trianggoro, W., Rambe, R.S., Ramadhan, F. and Arief, 2011. *Implementasi Berbagai Algoritme Neural Network Dan Wavelet Pada Field Programmable Gate Array*.

Jong, J., 2011. Predicting Rating with Sentiment Analysis. [Online] tersedia di: https//scholar.google.co.id/ [diakses 2 September 2018]

Kohavi, R., 2015. A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection. *Computer Science.*

Librian, A., 2017. High quality stemmer library for Indonesian Language (Bahasa). [Online] tersedia di: https://github.xom/sastrawi/sastrawi [diakses 12 November 2018].

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

Nurjannah, M., Hamdani, H. and Astuti, I.F., 2016. Penerapan Algoritme Term Frequency-Inverse Document Frequency (TF-IDF) untuk Text Mining. *Jurnal Informatika Mulawarman (JIM)*, 8(3), pp.110–113.

Rausanfita, A., Adikara, P., & Adinugroho, S. Analisis Sentimen Twitter Menggunakan Ensemble Feature dan Metode Extreme Learning Machine (ELM) (Studi Kasus: Samsung Indonesia). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 12, p. 6409-6417, agu. 2018. ISSN 2548-964X. Tersedia pada: <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/3612>. Tanggal Akses: 14 apr. 2019

Srimuang, W, Intarashothonchun & S, 2015. Classification Model of Network Intrusion Using Weighted Extreme Learning Machine. *International Joint*

Conference on Computer Scine And Software Engineering (JCSSE), Volume 12.

Sun, Z.-L., Choi, T.-M., Au, K.-F. and Yu, Y., 2008. Sales forecasting using extreme learning machine with applications in fashion retailing. *Decision Support Systems*, [online] 46(1), pp.411–419. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0167923608001371>.

You, Z.H., Li, S., Gao, X., Luo, X. and Ji, Z., 2014. Large-scale protein-protein interactions detection by integrating big biosensing data with computational model. BioMed Research International, 2014, pp.28–30.