PENENTUAN KUALITAS AIR SUNGAI MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Alvia Nur Azizah
NIM:125150201111010

INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PENENTUAN KUALITAS AIR SUNGAI MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Alvia Nur Azizah
NIM: 125150201111010

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Juni 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Edy Santoso, S.Si, M.Kom
NIK: 201201 850719 1 001	NIP: 19740414 200312 1 004

Mengetahui,
Pjs. Ketua Program Studi Informatika/Ilmu Komputer

Issa Arwani, S.Kom, M.Sc
NIP: 19830922 201212 1 003

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 pustaka.

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, 30 Juni 2016

Alvia Nur Azizah

125150201111010

KATA PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi – Nya karena atas rahmat dan hidayahNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "PENETUAN KUALITAS AIR SUNGAI MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan – bantuan baik lahir maupun batin selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada:

1. Imam Cholissodin, S.Si, M.Kom dan Edy Santoso, S.Si, M.Kom selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan bijaksana mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Dr. Eng. Riyanto Haribowo, ST, MT selaku dosen pakar yang membantu dalam analisa objek penelitian penulis.
3. Indriati, S.T, M.T selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
4. Wayan Firdaus M, Ph.D, Ir. Heru Nurwasito, M.Kom, Drs. Marji, M.T, dan selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Fakultas Ilmu Komputer Universitas Brawijaya.
5. Issa Arwani, S.Kom, M.Sc selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya.
6. Ibunda Nurmalistina, Bapak tersayang Alm. Agus Siswadi, Nenek Wirah, Abd. Harris, S.Pd, Syamsul Arifin, Abd. Rahem, S.Pd, Ayah Ikhsan dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabaran dalam membesarkan dan mendidik penulis, serta tiada henti – hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
7. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis. Khususnya Wayan Firdaus M, P.Hd yang telah merubah pandangan saya untuk menghargai sebuah proses.
8. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Albilaga Linggra Pradana, S.Kom yang telah membantu proses implementasi program skiripsi ini, Ryan Budi, S.Kom yang telah membantu pengumpulan data, M Faizal Sukma Dika, S.Kom, Rani Kurnia, S.Kom, Abdul Khoir, S.Kom, Andhica P, S.Kom, dan seluruh anggota kelas E Informatika 2012, terima kasih dukungan dan semangat yang selalu diberikan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya. Sahabat – sahabat penulis Angkatan 2012 Teknik Informatika Alpha dan

semua sahabat di luar sana, terima kasih atas segala bantuannya selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya.

10. Teman – teman Kos Edelweiss Jalan Terusan Ambarawa No. 35 Malang, Aisyah dan Jayanti yang telah memberikan semangat dan dukungan kepada penulis selama pengerjaan skripsi di luar kampus.
11. Teman – teman D'Viceroy yang memberi dukungan kepada penulis dan seluruh teman – teman yang tidak disebutkan satu persatu telah memberika semangat kepada penulis.
12. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat. Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga skripsi ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 30 Juni 2016

Alvia Nur Azizah
alvia.nura@gmail.com

ABSTRAK

Alvia Nur Azizah.2016: PENENTUAN KUALITAS AIR SUNGAI MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing: Imam Cholissodin, S.Si, M.Kom dan Edy Santoso, S.Si, M.Kom.

Air merupakan salah satu sumber daya alam yang memiliki fungsi sangat penting bagi kehidupan dan perikehidupan manusia. Sungai sebagai saluran utama pengalir air dari hulu ke hilir, memiliki banyak aktivitas domestik dan industri di sepanjang alirannya. Dinamika aliran tersebut menimbulkan perubahan kualitas dan kuantitas sungai secara signifikan. Kualitas air dijaga dengan melakukan analisis kualitas air sungai. Penggunaan sistem cerdas dirancang untuk mempermudah penentuan kualitas air secara komputasi. *Input* yang dibutuhkan adalah parameter uji kualitas air yang terdiri dari parameter fisika dan parameter kimia. Proses analisa kualitas air oleh sistem dilakukan dengan metode *extreme learning machine* (ELM). Implementasi metode ELM tidak memiliki acuan perhitungan terhadap aturan maupun ketentuan yang ada pada metode STORET melainkan membandingkan hasil penentuan keduanya, sehingga apabila dalam waktu tertentu terjadi perubahan perhitungan maupun ketentuan pada metode STORET tidak mempengaruhi perhitungan yang ada pada metode ELM. Metode ELM digunakan untuk menentukan kualitas air sungai kedalam 4 (empat) kelas yaitu memenuhi baku mutu (kondisi baik), tercemar ringan, tercemar sedang, dan tercemar berat. Hasil dari skenario pengujian didapatkan tingkat akurasi antara hasil perhitungan metode ELM dengan hasil diagnosa pakar menggunakan metode STORET sebesar 87,97%.

ABSTRACT

Alvia Nur Azizah.2016: ***DETERMINING THE QUALITY OF RIVER WATER USING EXTREME LEARNING MACHINE METHOD*.**

***Advisor*: Imam Cholissodin, S.Si, M.Kom dan Edy Santoso, S.Si, M.Kom.**

Water is a natural resource that has a very important function for life and human life. Diverter river as the main channel of water from upstream to downstream, has a lot of domestic and industrial activity along the stream. The flow dynamics cause changes in the quality and quantity of the river significantly. Water quality is maintained by analyzing water quality. The use of intelligent systems designed to facilitate the determination of water quality computing. Input is needed is a parameter water quality test that consists of physical parameters and chemical parameters. Water quality analysis process by the system is done by the method of extreme learning machine (ELM). ELM method implementations do not have a reference to the calculation of the rules and conditions contained in STORET methods but rather compare the results of the determination of both, so if within a certain time there is a change in the method of calculation as well as the provisions do not affect the calculation STORET that of the method of ELM. ELM method used to determine the quality of river water into the 4 (four) classes which meet quality standards (good condition), lightly polluted, medium polluted and heavily polluted. The results of the testing scenarios between the results obtained accuracy rate calculation method of ELM with expert diagnosis using methods STORET of 87.97%.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Sungai merupakan salah satu sumber daya alam yang mencukupi hajat orang banyak. Sungai juga menjadi sumber kebutuhan bagi manusia menjalankan aktifitas sehari – hari, seperti mengairi sawah, menggerakkan generator bagi perusahaan listrik, dan sebagai kebutuhan rumah tangga bagi masyarakat yang menggantungkan kebutuhan air pada sungai. Air sungai tidak lepas dari pencemaran yang merupakan hasil dari aktifitas industri – industri rumah tangga maupun industri besar.

Salah satu contoh waduk sutami merupakan waduk terbesar di Jawa Timur yang terletak di Desa Karangkates, Kecamatan Sumber Pucung, Kabupaten Malang. Waduk ini dibuat untuk menampung air untuk irigasi daerah hilir saat kemarau hingga mencapai 24 m per detik. Selain itu, Waduk ini memberikan pasokan ke sawah mencapai 34.000 hektar. Volume air yang bias ditampung Waduk ini mencapai 343.000.000 m^3(Juantari, 2013).

Keberadaan lahan pemukiman di Daerah Aliran Sungai (DAS) mengakibatkan berbagai macam masalah, mulai dari terjadinya banjir, berkurangnya ketersediaan air yang diakibatkan semakin sempitnya lebar sungai hingga terjadinya pencemaran air yang mengakibatkan penurunan kualitas air sungai, dimana sebagian besar air sungai digunakan untuk menopang kehidupan masyarakat sekitar Daerah Aliran Sungai. Pencemaran sungai dari adanya permukiman dapat berasal dari buangan air rumah tangga, padatan berupa sampah yang dibuang ke sungai, air cucian kamar mandi, buangan tinja. Keempat hal tersebut akanmempengaruhi tingkat kandungan *Biological Oxygen Demand* (BOD), *Chemical Oxygen Demand (*COD) serta bakteri *E. Coli* dalam sungai.

Penurunan kualitas air akan menurunkan dayaguna, hasil guna, produktivitas, daya dukung dan daya tampung dari sumberdaya air yang pada akhirnya akan menurunkan kekayaan sumberdaya air. Untuk menjaga kualitas air agar tetap pada kondisi alamiahnya, perlu dilakukan pengelolaan dan pengendalian pencemaran air secara bijaksana. Hal ini dibutuhkan upaya pemantauan dan pengendalian pencemaran terhadap air sungai. Upaya yang dilakukan yaitu melakukan pengukuran dan analisis terhadap air sungai tentang status mutu air sebagaimana yang ditetapkan dalam Peraturan Pemerintah no 82 tahun 2001.

Kementrian Lingkungan Hidup mengeluarkan keputusan Nomor: 115 Tahun 2003 tentang penentuan mutu air dengan Metode STORET atau Metode Indeks Pencemaran.Penentuankualitas air sungai dengan Metode STORET masihdilakukan secara manual dengan cara menghitung satu – persatu data parameter sehingga membutuhkan waktu yang lama dan biaya pengujian yang besar (Annisah, 2011). Oleh sebab itu, penulis mengajukan sebuah penelitian menggunakan metode *Extreme Learning Machine* (ELM) untuk memudahkan perhitungan kualitas air sungai dengan *input* berupa parameter air meliputi: TSS, BOD, COD, DO, pH, Fenol, Minyak dan Lemak. Penerapan algoritma ELM, dapat

memberikan solusi bagi pengguna dalam membantu proses terkait dengan penentuan kualitas air sungai.

Beberapa penelitian sebelumnya yang berkaitan dengan penentuan status mutu air telah dilakukan oleh Annisah (2011). Pada penelitian tersebut Annisah meneliti status mutu air sungai di waduk sutami dengan metode STORET menggunakan 9 (sembilan) parameter yang terdiri dari BOD,COD, DO, TSS, pH, Amonia (NH3_N), fenol, minyak dan lemak, dan sianida (CN). Hasil dari penelitian tersebut, status mutu air dibagi menjadi 4 (empat) kriteria yaitu memenuhi baku mutu (kondisi baik), tercemar ringan, tercemar sedang, dan tercemar berat.

Pada bidang teknologi terdapat metode jaringan saraf tiruan (JST) atau dalam bahasa Inggris disebut *artificial neural networ*k (ANN). Metode JST mengadopsi sistem pembelajaran pada otak manusia. Menurut Sun dkk (2008) metode JST lebih baik daripada metode-metode peramalan konvensional lainnya. Pada penelitian ini metode JST yang digunakan yaitu *extreme learning machine* (ELM). ELM pertama kali diperkenalkan oleh Huang (2004) merupakan metode *feedforward* dengan *single hiden layer* atau disebut dengan istilah *single hidden layer feedforward neural networ*k (SLFNs) memiliki kemampuan *learning speed* yang baik.

ELM juga diterapkan dalam bidang manajemen rumah sakit seperti yang dilakukan oleh Fardani (2015). Dalam penelitiannya Fardani menerapkan metode ELM kedalam sistem pendukung keputusan dalam menentukan jumlah pasien yang berkunjung di rumah sakit. Penelitian terkait juga dilakukan Cao dan Xiong (2014) tentang klasifikasi protein sekuensial menggunakan ELM. Dalam penelitiannya, Cao dan Xiong membandingkan hasil penerapan ELM dengan metode kombinasi ELM lainnya diperoleh peningkatan yang signifikan pada bagian *training speed* data.

Berdasarkan paparan yang telah dijelaskan diatas, penulis mengusulkan penelitian kualitas air sungai menggunakan ELM sehingga diperoleh hasil dengan tingkat akurasi yang baik agar dapat membantu pemerintah dalam menanggulangi tingkat pencemaran air sungai. Pada penelitian ini, untuk menentuan kualitas air sungai menggunakan 7 (tujuh) parameter *input* terdiri dari residu tersuspensi (TSS), BOD, COD, DO, pH, Fenol, serta Minyak dan Lemak. ELM digunakan untuk menentukan kualitas air sungai kedalam 4 (empat) kelas yaitu memenuhi baku mutu (kondisi baik), tercemar ringan, tercemar sedang, dan tercemar berat.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah penulis paparkan sebelumnya diperoleh rumusan masalah dalam penelitian ini sebagai berikut.

1. Bagaimana menerapkan metode *Extreme Learning Machine* (ELM) dalam menentukan kualitas air sungai?
2. Bagaimana tingkat akurasi dari hasil penerapan metode *Extreme Learning Machine* (ELM) dalam menentukan kualitas air sungai?

1.3 Tujuan

Adapun tujuan dari penelitian yang dilakukan penulis.

1. Menerapkan metode ELM dalam penentuan Kualitas Air Sungai.
2. Menguji tingkat akurasi hasil penerapan metode ELM dalam penentuan Kualitas Air Sungai.

1.4 Manfaat

Manfaat dari penelitian yang dilakukan penulis antara lain:

1. Bagi Penulis

Dapat memahami penerapan Algortima ELM untuk menentukan kualitas air sungai di waduk sutami, dan hasil dari penelitian yang dilakukan dapat memberikan kontribusi dalam penelitian selanjutnya berkaitan dengan topik yang serupa.

2. Bagi Pengguna

1. Mempermudah perhitungan pada tingkat pencemaran sehingga pemerintah melakukan tindakan penanggulangan terhadap tingkat pencemaran air sungai.
2. Memberikan informasi sehingga mengurangi aktifitas rumah tangga maupun industri yang menyebabkan tingkat pencemaran air sungai tinggi.

1.5 Batasan masalah

1. Fitur data yang digunakan pada penelitian ini adalah TSS, BOD, COD, DO, pH, Fenol, Minyak dan Lemak.

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam tugas akhir ini sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, sistematika pembahasan, dan jadwal penelitian yang berlangsung dalam menentukan kualitas air sungai menggunakan ELM.

BAB II LANDASAN KEPUSTAKAAN

Landasan kepustakaan menjelaskan tentang kajian pustaka terkait dengan penelitian yang telah ada seperti penelitian tentang penentuan kualitas air sungai menggunakan ELM. Dasar teori yang diperlukan untuk mendukung penelitian ini adalah Air Sungai, Jaringan Saraf Tiruan (JST), *Extreme Learning Machine* (ELM).

BAB III METODOLOGI

Membahas tentang metode yang digunakan dalam penelitian tentang penentuan kualitas air sungai menggunakan ELM dan langkah kerja yang akan dilakukan dalam penulisan laporan akhir penelitian.

BAB IV PERANCANGAN SISTEM

Membahas tentang bagaimana merancang penelitian yang dilakukan berkaitan dengan sebuah komputasi yang akan digunakan dalam penentuan kualitas air sungai dengan metode ELM.

BAB V IMPLEMENTASI

Membahas tentang implementasi metode ELM yang digunakan kedalam sebuah komputasi berupa program untuk menentukan kualitas air sungai.

BAB VI PEGUJIAN DAN PEMBAHASAN

Membahas tentang hasil pengujian fungsional dan pengujian akurasi terhadap program komputasi menggunakan ELM untuk menentukan kualitas air sungai.

BAB VII PENUTUP

Memuat kesimpulan yang diperoleh dari penelitian tentang penentuan kualitas air sungai menggunakan ELM yang dikembangkan dalam skripsi ini serta saran – saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Juga terdapat kajian pustaka yang menjelaskan secara umum penelitian-penelitian terdahulu yang berhubungan dengan topik skripsi dan menunjukkan persamaan dan perbedaan skripsi ini terhadap penelitian terdahulu yang dituliskan.

2.1 Kajian Pustaka

Kajian pustaka berisi penelitian-penelitian sebelumnya yang terkait atau memiliki kesesuaian dengan penelitian penulis. Kajian pustaka tersebut dijelaskan pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No	Judul	Obyek	Metode	Hasil
1	*Application of extreme learning machine for estimation of wind speed distribution*	Kecepatan ditribusi angin	Algoritma *Extreme Learning Machine*	Penelitian ini menggunakan metode ELM untuk memperoleh nilai bentuk (k) dan skala (c) kecepatan angin yang diambil dari persentase nilai *error* terkecil sebesar .
2	*Sales Forecasting using Extreme Learning Machine with Application in Fashion Retailing.Elsevier Decision Support Systems*	Penjualan *fashionretail* (penjualan eceran)	Algoritma *Extreme Learning Machine*	Penelitian ini menerapkan ELM untuk menyelidiki hubungan antara jumlah penjualan dan beberapa faktor penting yang mempengaruhi permintaan (seperti faktor desain). Hasil percobaan menunjukkan bahwa metode ELM mengungguli beberapa metode peramalan penjualan lainnya.
3	*Protein Sequence Classification with Improved Extreme*	Protein Sekuensial	Algoritma *Extreme Learning Machine*	Penelitian ini menggunakan ELM sebagai metode klasifikasi protein sekuensial dan membandingkan dengan

	Learning Machine Algorithms			metode lain yang dikombinasikan dengan ELM. Hasilnya, metode ELM memiliki *learning speed* yang baik.
4	Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode *Extreme Learning Machine* (Studi Kasus: Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto)	Peramalan Jumlah Kunjungan Pasien	Algoritma *Extreme Learning Machine*	Jumlah kunjungan pasien dalam sebuah rumah sakit sering kali berubah – ubah, hal ini tentu berpengaruh dengan pelayanan rumah sakit. Dengan dibuat sebuah sistem peramalan diharapkan bisa memberikan prediksi kunjungan pasien dalam kurun waktu tertentu. Metode ELM sangat cocok untuk peramalan karena memiliki akurasi yang baik dengan tingkat *error* yang relatif kecil.

Sumber : (Shamshirband dkk, 2015), (Sun dkk, 2008), (Cao dan Xiong, 2014), (Fardani, 2015).

Berdasarkan paparan dari referensi diatas, terdapat hasil yang relevan dengan penelitian skripsi yang akan dilakukan oleh penulis. Penelitian yang dilakukan oleh Shamshirband dkk (2015) mengenai distribusi angin. Energi angin merupakan salah satu sumber daya yang paling tepat dan menarik yang dapat digunakan untuk memasok kebutuhan energi dan memberikan dasar energi yang berkelanjutan bagi negara-negara. Penelitian ini mencari nilai prediksi yang tepat dari dua parameter fungsi distribusi angin yang digunakan untuk menghitung potensi tenaga angin, yaitu bentuk (k) dan skala (c) dari fungsi Weibull. Hasilnya, ELM sangat menjanjikan sebagai metode alternatif untuk mereproduksi bentuk dan skala parameter fungsi Weibull setelah pelatihan dan kemudian mewakili distribusi kecepatan angin. Selain itu, ELM yang dikembangkan memiliki banyak fitur menarik dan luar biasa yang membuatnya dibedakan dari algoritma berdasarkan *gradien-populer* tradisional belajar untuk jaringan saraf *feedforward*. ELM jauh lebih cepat dalam kecepatan belajar dibandingkan dengan algoritma pembelajaran jaringan *feedforward* tradisional seperti algoritma *backpropagation*. Kedua, tidak seperti algoritma pembelajaran tradisional, algoritma ELM mampu mencapai kesalahan pelatihan terkecil dan juga norma bobot.

Penelitian kedua yang dilakukan oleh Sun dkk (2008) mengenai peramalan penjualan untuk mengetahui hubungan antara jumlah penjualan dan beberapa faktor penting yang mempengaruhi permintaan (dalam kasus tersebut

faktor desain). Penelitian ini menggunakan metode Jaringan Saraf Tiruan yaitu ELM. Hasil percobaan menunjukkan bahwa metode ELM lebih mengungguli beberapa metode JST yang digunakan dalam peramalan penjualan yang seperti pada *back*propagation.

Penelitian ketiga yaitu dilakukan oleh Cao dan Xiong (2014) membahas pengelompokan protein sekuensial. Klasifikasikan urutan protein anggota menjadi protein dari suatu kategori (superfamily) akan bermanfaat dalam beberapa analisis molekuler dalam kategori (superfamily) tertentu, bukan analisis pada semua urutan protein anggota individu. Umumnya, dua sekuens protein diklasifikasikan ke dalam kategori yang sama jika pola fitur mereka diekstraksi menggunakan algoritma dengan urutan selaras menunjukkan homologi tinggi. ELM adalah metode tercepat dan fase pelatihan dari dataset urutan protein membutuhkan waktu kurang dari 1 detik. Namun, metode BP konvensional membutuhkan lebih dari 1.000 detik ketika menerapkan *kernel linear* dan *kernel Gaussian* dan lebih dari 1 detik bila menggunakan *kernel sigmoid*. Waktu pelatihan biaya dengan SVM lebih dari detik untuk ketiga kernel tersebut. Meskipun algoritma ELM didimodifikasi menjadi VOP-ELM dan OP-ELM, waktu pelatihan meningkat sangat signifikan karena mencari jumlah *hidden neuron* yang optimal.

Penelitian terakhir dilakukan oleh Fardani (2015) tentang ELM diterapkan dalam sistem pendukung keputusan peramalan kunjungan pasien rumah sakit. Dalam peneitian ini ELM digunakan dalam perhitungan dengan jumlah data 80% (463 data) dari total 579 data dan 20% (116 data) sisanya sebagai data *testing* yang kemudian di *normalisasi*. Didapatkan hasil optimal dengan nilai MSE sebesar 0.027 dengan *hidden layer* sebanyak 7 unit dan Epoch 500.

Berdasarkan penjelasan penelitian diatas, metode ELM efektif dalam penentuan prediksi dengan *hidden layer* mempercepat proses *learning speed* dan *input* bobot yang dipilih secara acak. Hal ini, menunjang penulis untuk melakukan penelitian serupa dengan menggunakan metode ELM dalam menentukan kualitas air sungai.

2.2 Air Sungai

Air adalah sumber daya alam yang dibutuhkan untuk hajat hidup orang banyak, bahkan oleh semua makhluk hidup. Oleh sebab itu, perlu dilindungi agar tetap bermanfaat bagi kehidupan. Hal ini berarti bahwa pemanfaatan air harus dilakukan secara bijak dengan mempertimbangkan kepentingan generasi sekarang dan yang akan datang. Namun, semakin pesatnya pembangunan, baik bidang pertanian, peternakan, industri dan lain – lain, serta laju pertumbuhan penduduk yang semakin pesat menyebabkan pemanfaatan air tidak lagi dilakukan dengan baik. Hal tersebut memberikan dampak buruk yang mempengaruhi sifat fisik dan sifat kimia air sehingga menurunkan kualitas air. Pengelolaan sumber daya air bertujuan menyediakan air dalam jumlah yang cukup dengan kualitas yang sesaui dengan peruntukannya.

Di dalam suatu sistem Daerah Aliran Sungai (DAS), sungai yang berfungsi sebagai wadah pengaliran air selalu berada di posisi paling rendah, sehingga tidak dapat dipisahkan dari kondisi Daerah Aliran Sungai (PP 38 Tahun 2011). Kualitas air sungai dipengaruhi oleh kualitas pasokan air yang berasal dari daerah tangkapan sedangkan kualitas pasokan airdari daerah tangkapan berkaitan dengan aktivitas manusia yang ada di dalamnya (Wiwoho, 2005). Perubahan kondisi kualitas air pada aliran sungai merupakan dampak dari buangan dari penggunaan lahan yang ada (Tafangenyasha dan Dzinomwa, 2005). Selain itu, berbagai aktivitas manusia dalam memenuhi kebutuhan hidupnya yang berasal dari kegiatan industri, rumah tangga, dan pertanian akan menghasilkan limbah yang memberi sumbangan pada penurunan kualitasair sungai (Suriawiria, 2003). Berbagai aktivitas penggunaan lahan di wilayah DAS seperti aktivitas permukiman, pertanian dan industri diperkirakan telah mempengaruhi kualitas air sungai (Agustiningsih, 2014).

Menurut PP. Nomor 82 Tahun 2001 tentang klasifikasi mutu air ditetapkan menjadi 4 yaitu:

Kelas I : air yang peruntukannya dapat digunakan untuk air baku air minum, dan atau peruntukan lain yang mempersyaratkan mutu air sama dengan kegunaan tersebut;

Kelas II : air yang peruntukannya dapat digunakan untuk prasarana/sarana rekreasi air, pembudidayaan ikan air tawar, peternakan, air untuk mengairi pertanaman, dan atau peruntukan lain yang mempersyaratkan mutu air sama dengan kegunaan tersebut;

Kelas III : air yang peruntukannya dapat digunakan untuk pembudidayaan air tawar, peternakan, air untuk mengairi pertanaman, dan atau peruntukan lain yang mempersyaratkan mutu air sama dengan kegunaan tersebut;

Kelas IV : air yang peruntukannya dapat digunakan untuk mengairi pertanaman dan atau peruntukan lain yang mempersyaratkan mutu air sama dengan kegunaan tersebut.

2.2.1 Mutu Air

Mutu air adalah suatu kondisi kualitas air yang diukur atau diuji berdasarkan parameter – parameter tertentu dan metode tertentu berdasarkan peraturan perundang-undangan yang berlaku. Status mutu air dapat mengindikasikan keadaan air dalam kondisi tercemar atau bebas dari pencemaran dalam waktu tertentu dengan membandingkan dengan baku mutu yang ditetapkan. Kementrian Lingkungan Hidup mengeluarkan keputusan Nomor : 115 Tahun 2003 tentang penentuan mutu air dengan Metode STORET atau Metode Indeks Pencemaran dengan 4 kriteria status mutu air yaitu, memenuhi baku mutu (kondisi baik), tercemar ringan, tercemar sedang, dan tercemar berat (Annisah, 2011). Contoh kondisi air sungai mulai dari air bersih hingga tercemar berat seperti pada Gambar 2.1, Gambar 2.2, Gambar 2.3 dan Gambar 2.4.

(a)

(b)

(c)

(d)

Gambar 2.1 Contoh Fisik Air Sungai, yaitu (a) Memenuhi Baku Mutu, (b) Tercemar Ringan, (c) Tercemar Sedang, (d) Tercemar Berat

2.2.2 Parameter Fisika-Kimia untuk Uji Kualitas Air

Parameter pencemar air merupakan indikator yang memberi petunjuk terjadinya pencemaran air. Dengan adanya indikator ini pencemaran dapat diatasi sedini mungkin atau paling tidak sedikit dikurangi. Pada penelitian ini digunakan acuan Keputusan Menteri Negara Lingkungan Hidup Nomor : 115 tahun 2003 tentang Pedoman Penentuan Status Mutu Air sebagai standar parameter mutu air. Sedangkan, Baku Mutu Air menggunakan acuan Peraturan Pemerintah RI No. 82 Tahun 2001 tentang Pengelolaan Kualitas Air dan Pengendalian Pencemaran. Beberapa parameter fisika-kimia untuk menguji kualitas air sebagaimana diuraikan dibawah ini:

a. *Total Suspended Solid* (TSS)

Total Suspended Solid (TSS) merupakan zat padat yang terdapat dalam sespensi, dapat dibedakan menurut ukurannya sebagai partikel tersuspensi koloid (partikel koloid) dan partikel tersuspensi biasa (pertikel tersuspensi). *Total Suspended Solid* (TSS) yaitu jumlah berat dalam satuan mg/l kering lumpur yang ada didalam air limbah setelah mengalami proses penyaringan dengan membran berukuran 0.45 µm. Adanya padatan – padatan ini menyebabkan kekeruhan air, padatan ini tidak larut dan tidak mengendap

secara langsun. Padatan tersuspensi terdiri dari partikel – partikel yang ukuran maupun beratnya lebih kecil daripada air.

Kadar TSS yang tinggi akan menghalangi masuknya sinar matahari ke dalam air, sehingga akan mengganggu proses fotosintesis yang menyebabkan turunnya kadar oksigen terlarut yang dilepas kedalam air oleh tanaman. Apabila sinar matahari terhalangi dari dasar tanaman, maka tanaman akan berhenti memproduksi oksigen dan akan mati. TSS juga menyebabkan penurunan kejernihan dalam air Annisah (2011). Kekeruhan air yang disebabkan oleh zat padat tersuspensi bersifat anorganik dan organik. Zat anorganik biasanya berasal dari lapukan batuan dan logam, sedangkan zat organik berasal dari lapukan tanaman atau hewan.

b. Kebutuhan Oksigen Biologi (BOD)

BOD merupakan banyaknya oksigen dalam ppm atau mg/l yang diperlukan untuk menguraikan benda organik oleh bakteri, sehingga limbah tersebut menjadi jernih kembali. Apabila dalam air banyak mengandung bahan – bahan organik, maka semakin banyak pula oksigen yang diperlukan bakteri untuk menguraikan zat organik, hal ini dapat mengakibatkan menurunnya kadar oksigen dalam air. Semakin besar kadar BOD menunjukkan tingginya tingkat kekotoran air. Pengukuran BOD penting karena merupakan parameter untuk menentukan daya cemar air.

c. Kebutuhan Oksigen Kimiawi (COD)

COD adalah banyaknya oksigen dalam ppm atau milligram per liter (mg/l) yang dibutuhkan dalam kondisi khusus untuk menguraikan benda organik secara kimiawi.

d. Oksigen Terlarut (DO)

DO adalah banyaknya oksigen yang terkandung didalam air dan diukur dalam satuan milligram per liter (mg/l). Oksigen yang terlarut ini dipergunakan sebagai tanda derajat pengotoran limbah yang ada. Semakin besar kadar oksigen yang terlarut, maka derajat kekotoran yang ada semakin tinggi.

e. Derajat Keasaman (pH)

Konsentrasi ion hidrogen adalah ukuran mutu dari air maupun air limbah. Adapun kadar yang baik yaitu masih memungkinkan kehidupan di dalam air yang berlangsung dengan baik. Air limbah dengan konsentrasi keasaman yang tidak netral akan menyulitkan proses biologis, sehingga mengganggu proses penjernihan. Kondisi yang baik bagi air minum dan air limbah adalah netral (7). Semakin kecil derajat keasamannya maka kondisi air akan semakin asam, sebaliknya semakin besar derajat keasamannya maka kondisi air akan melampaui batas netral yaitu basa.

f. Fenol

Fenol merupakan penyebab timbulnya rasa yang ada didalam air minum maupun air limbah. Fenol ini disebabkan oleh limbah industri dan apabila konsentrasinya mencapai 500 mg/l masih dapat dioksidasi melalui proses biologis.

g. Minyak dan Lemak

Minyak dan lemak merupakan komponen utama bahan makanan yang juga banyak terkandung didalam air. Apabila komponen ini tidak dihilangkan sebelum masuk ke pembuangan, maka dapat mempengaruhi kehidupan yang ada di permukaan air dan menimbulkan lapisan tipis di permukaan sehingga membentuk selaput.

2.3 Metode STORET

Metode STORET merupakan salah satu metode untuk menentukan status mutu air yang umum digunakan. Dengan metode STORET ini dapat diketahui parameter-parameter yang telah memenuhi atau melampaui baku mutu air. Secara prinsip metode STORET adalah membandingkan antara data kualitas air dengan baku mutu air yang disesuaikan dengan peruntukannya guna menentukan status mutu air. Cara untuk menentukan status mutu air berdasarkan peraturan pemerintah yang telah ditetapkan adalah dengan menggunakan sistem nilai dari "US-EPA (*Environmental Protection Agency*)" dengan mengkelasifikasikan mutu air dalam empat kelas pada Tabel 2.2, yaitu:

Tabel 2.2 Klasifikasi Status Mutu Air Menurut "US-EPA"

No	Kelas	Kategori	Skor	Keterangan
1.	Kelas A	Baik Sekali	0	Memenuhi baku mutu
2.	Kelas B	Baik	-1 s/d -10	Tercemar Ringan
3.	Kelas C	Sedang	-11 s/d -30	Tercemar Sedang
4.	Kelas D	Buruk	≥ -31	Tercemar Berat

Adapun langkah – langkah penggunaan metode STORET akan dipaparkan dibawah ini :

1. Lakukan pengumpulan data kualitas air dan debit air secara periodik sehingga membentuk data dari waktu kewaktu (*time series* data).
2. Bandingkan data hasil pengukuran dari masing-masing parameter air dengan nilai baku mutu yang sesuaidengan kelas air.
3. Jika hasil pengukuran memenuhi nilai baku mutu air (hasil pengukuran < baku mutu) maka diberi skor 0.
4. Jika hasil pengukuran tidak memenuhi nilai baku mutu air (hasil pengukuran > baku mutu), maka diberi skor pada Tabel 2.3:

Tabel 2.3 Nilai Baku Mutu Air

Jumlah yang Digunakan[1)]	Nilai	Parameter		
		Fisika	Kimia	Biologi
< 10	Maksimum	-1	-2	-3
	Minimum	-1	-2	-3
	Rata – rata	-3	-6	-9
≥ 10	Maksimum	-2	-4	-6
	Minimum	-2	-4	-6
	Rata – rata	-6	-12	-18

Sumber : keputusan menteri lingkungan hidup nomor 115 tahun 2003
Catatan : 1) jumlah parameter yang digunakan untuk penentuan status mutu air.

5. Jumlah negatif dari seluruh parameter dihitung dan ditentukan status mutunya dari jumlah skor yang didapat dengan menggunakan sistem nilai.
6. Menghitung total jumlah negatif dari seluruh parameter dan menentukan status mutunya dari jumlah skor yang didapat dengan menggunakan sistem nilai dari US-EPA.

Contoh perhitungan STORET:

Penetuan Status Mutu Air dengan Metode STORET Sungai Bedog Lokasi Pemantauan Jembatan Sempor, Sleman

No	Parameter	Satuan	Baku Mutu	Hasil Pengukuran		Max	Min	Rata-rata	skor
				Bulan I	Bulan II				
1.	FISIKA								
a.	TSS	mg/L	50	14	14	14	14	14	0
2.	KIMIA								
a.	anorganik								
3.	BOD	mg/L	3	12	4.1	12	4.1	8.05	-10
4.	COD	mg/L	25	20.2	10.3	20.2	10.3	15.25	0
5.	DO	mg/L	>4	6.2	7	7	6.2	6.6	0
6.	Sianida(CN)	mg/L	0.02	0.001	0.004	0.004	0.001	0.0025	0
7.	pH	mg/L	6-8.5	7.3	6.8	7.3	6.8	7.05	0
8.	NH3_N	mg/L	0.5	0.44	0.01	0.44	0.01	0.225	0
b.	organik								
9.	Fenol	mg/L	0.001	0.0001	0.0001	0.0001	0.0001	0.0001	0
10.	Minyak dan Lemak	mg/L	1	1	0	1	0	0.5	0

TOTAL	-10

Cara pemberian skor untuk setiap parameter adalah sebagai berikut (contoh, untuk BOD):

a. BOD merupakan parameter kimia maka gunakan skor untuk parameter kimia.
b. Baku mutu air untuk BOD adalam 3 mg/L.
c. Kadar BOD maksimum hasil pengukuran adalah 12 mg/L, hal ini berarti kadar BOD melebihi baku mutunya. Maka, skor untuk nilai maksimum adalah -2.
d. Kadar BOD minimum adalah 4.1 mg/L, hal ini berarti kadar BOD melebihi baku mutunya. Maka, skor untuk nilai minimum adalah -2.
e. Kadar BOD rata-rata hasil pengukuran adalah 8.05 mg/L, hal ini berarti kadar BOD melebihi baku mutunya. Maka, skor untuk nilai rata-rata adalah -6.
f. Jumlahkan seluruh skor untuk nilai maksimum, minimum, dan rata-rata. Skor BOD pada contoh ini adalah -10.
g. Lakukan langkah yang sama untuk tiap parameter, apabila tidak ada baku mutunya untuk parameter tertentu, maka tidak perlu dilakukan perhitungan.
h. Kemudian, jumlahkan seluruh skor parameter. Pada contoh ini total skor keseluruhan adalah -10, hal ini berarti air sungai tersebut tergolong Tercemar Ringan (Kelas B).

Metode STORET masih melakukan perhitungan manual yaitu dengan menghitung satu persatu setiap parameter kemudian menjumlahkan keseluruhan hasil perhitungannya, selain itu pula waktu yang dibutuhkan lama.

2.4 Jaringan Saraf Tiruan (JST)

2.4.1 Pengertian Jaringan Saraf Tiruan (JST)

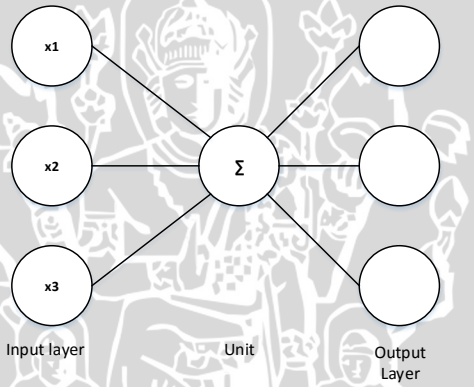
JST atau dalam bahasa Inggris *Artifcial Neural Networ*k (ANN) adalah upaya untuk memodelkan pemrosesan informasi berdasarkan kemampuan sistem saraf biologis yang ada pada manusia (Fardani, 2015). Jadi JST merupakan jaringan saraf biologis dipandang dari sudutpandang pengolahan informasi. Hal ini dimungkinkan untuk merancang model yang dapat disimulasikan dan dianalisis. Dalam JST neuron – neurondi kelompokkan dalam lapisan – lapisan (layer). Umumnya, neuron-neuron yang terletak pada layer yang sama akan memiliki keadaan yang sama. Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada setiap layer yang sama, neuron-neuron akan memiliki fungsi aktivasi yang sama.

Dalam (Siang, 2005) dijelaskan bahwa arsitektur jaringan yang sering dipakai dalam JST adalah *single layer network* (Jaringan Layar Tunggal) dan *multi layer network* (Jaringan Layar Banyak). Selain itu diterangkan juga

tentang fungsi aktivasi yang merupakan aturan yang memetakan penjumlahan input elemen pemroses terhadap outputnya. Fungsi ini adalah fungsi umum yang akan digunakan untuk menentukan keluaran suatu neuron. Tujuan lain dari fungsi ini adalah untuk memodifikasi output kedalam rentang nilai tertentu. Fungsi-fungsi aktivasi yang biasanya digunakan dalam sistem Jaringan Saraf Tiruan: Fungsi *Step Biner*, Fungsi *Sigmoid Biner* dan Fungsi *Sigmoid Bipolar*.

2.4.2 Neuron Jaringan Saraf Tiruan

Dalam otak manusia terdapat jutaan neuron, begitu pula pada jaringan saraf tiruan terdiri dari neuron yang saling terhubung satu sama lain. Dalam hal ini, neuron melakukan proses mengolah data yang akan diteruskan pada neuron – neuron lainnya berupa keluaran yang akan di proses selanjutnya. Hal ini merupakan neuron adalah unit dasar yang melakukan mengolah informasi pada jaringan saraf tiruan. Struktur neuron pada jaringan saraf tiruan di gambarkan pada Gambar 2.5 (Hermawan, 2006).



Gambar 2.2 Struktur Unit Neuron

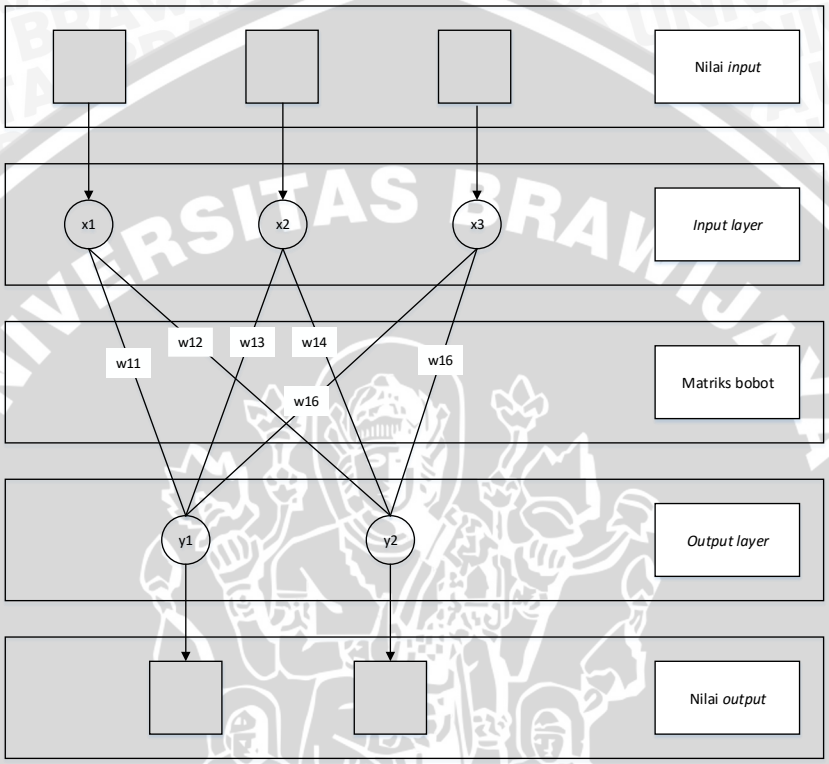
Gambar 2.5 merupakan struktur dari unit neuron pada jaringan saraf tiruan yang terdiri dari 3 bagian, yaitu *input*, unit neuron, dan *output*. *Input* (masukan) diproses oleh unit neuron (pengelola) dari setiap masukan yang berbeda (x_n). Setiap masukan memiliki *connector* (penghubung) yang memiliki besaran nilai (bobot) dinotasikan dengan simbol w_n pada *input* dan pada *output* sebagai *connector* untuk mengirimkan hasl keluaran. Unit neuron melakukan proses matematis dari setiap masukan disertai bobot masing – masing untuk menghasilkan keluaran. Hasil *output* akan diteruskan melalui *connector* pada *output*.

2.4.3 Arsitektur Jaringan Saraf Tiruan

Jaringan saraf tiruan memiliki suatu aturan yang pada dasarnya sama (*general rule*) pada semua model jaringan. Perbedaan arsitektur jaringan akan menentukan tingkat keberhasilan pemecahan masalah (Hermawan, 2006).

1. Jaringan dengan lapisan tunggal (*single layer networ*k)

*Single layer networ*k dirancang oleh Widrow dan Holf di tahun 1960. Struktur jaringan sangat sederhana yaitu terdiri dari *layer input* dan *output*. *Layer input* bagian yang menerima masukan yang akan di proses, sedangkan *layer output* sebagai keluaran berupa respon dari hasil masukan. Keduanya dihubungkan oleh *layer* bobot yang akan memproses masukan dari *layer input* dan hasilnya akan diteruskan ke *layer output*.

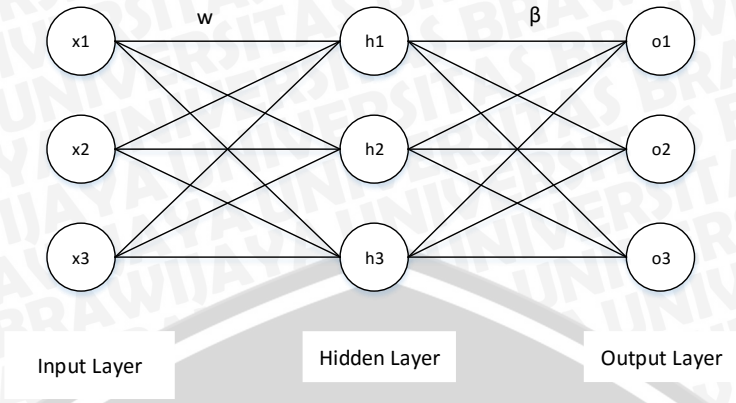


Gambar 2.3 Struktur *Single Layer Networ*k

Gambar 2.6 *input layer* dihubungkan oleh matriks bobot dengan masing – masing nilai bobot yang bersesuaian. Data akan diterima oleh neuron – neuron yang ada pada *input layer*. Seberapa besar hubungan antara neuron tersebut ditentukan oleh bobot yang bersesuaian di setiap neuron. Semua unit *input* dihubungkan pada setiap unit *output*.

2. Jaringan lapis banyak (*multilayer networ*k)

Sama seperi *single layer networ*k, *multilayer networ*k memiliki bagian unit tersembunyi (*hidden layer*). Banyaknya *hidden node* (unit tersembunyi) disesuaikan dengan kebutuhan dalam permasalahan yang berbeda. Secara umum, semakin kompleks jaringan semakin banyak pula jumlah *hidden node*, demikian pula dengan jumlah layernya.



Gambar 2.4 Struktur *Multi Layer Networ*k

Pada Gambar 2.7, terdapat lapisan input dengan banyaknya neuron input $(x_1, x_2, \dots, x_n)$. Lapisan tersembunyi ada satu dengan banyaknya neuron tersembunyi $(h_1, h_2, \dots, h_n)$. Dan lapisan output dengan banyaknya neuron $(o_1, o_2, \dots, o_n)$.). Bobot-bobot yang menghubungkan neuron input ke-n menuju neuron ke-*h* pada lapisan tersembunyi disimbolkan dengan *w*, sedangkan β adalah bobot-bobot dari neuron ke-*n* pada lapisan tersembunyi yang menuju ke-*o* pada lapisan output.

3. Jaringan kompetitif (*competitive layer networ*k)

Jaringan ini memiliki lapisan kompetitif. Lapisan kompetitif ini digunakan dalam perhitungan dengan data yang diperoleh sebelumnya. Secara prinsip lapisan ini membandingkan data yang baru dengan data sebelumnya.

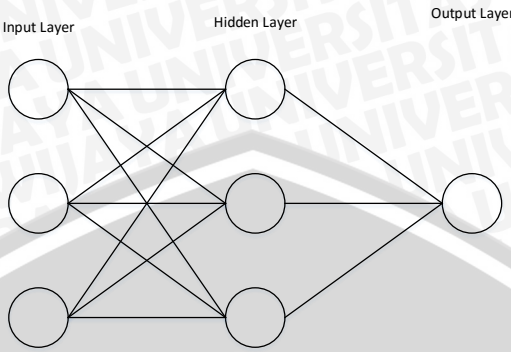
2.5 *Extreme Learning Machine* (ELM)

ELM merupakan salah satu metode baru dari segi pembelajaran yang ada pada JST. Metode ini pertama kali diperkenalkan oleh Huang dkk (2004). ELM merupakan jaringan saraf tiruan *feedforward* dengan *single hidden layer* atau biasa disebut dengan *Single Hidden Layer Feedforward neural Networks* (SLFNs) (Sun dkk, 2008). Metode pembelajaran ELM dibuat untuk mengatasi kelemahan-kelemahan dari jaringan saraf tiruan *feedforward* terutama dalam hal *learning speed*.

Menurut Huang dkk (2004), JST *feedforward* masih memiliki kelemahan dalam *learning speed* karena semua parameter pada jaringan ditentukan secara iterative dengan menggunakan metode pembelajaran tersebut.

Pada proses *learning* JST semua parameter harus ditentukan secara manual (Huang dkk,2005). Parameter yang dimaksud adalah *input weight* dan *bias*. Parameter – parameter tersebut juga saling berhubungan antara layer yang satu dengan yang lain, sehingga membutuhkan *learning speed* yang lama. Sedangkan pada ELM parameter-parameter – parameter tersebut dipilih secara random, sehingga ELM memiliki *learning speed* yang cepat dan mampu menghasilkan *good generalization performance*. Hal itu

menjadikan ELM seribu kali lebih cepat dari algoritma *feedforward* BP. Gambar 2.3 adalah struktur ELM.



Gambar 2.5 Struktur ELM

Struktur ELM terdiri dari 3 *layer*, yaitu *input layer*, *hidden layer*, dan *output layer*. Setiap *node* pada *input layer* terhubung dengan *hidden node* yang ada pada *hidden layer*. *Node* tersebut dihubungkan oleh bobot yang disebut bobot *input* dengan nilai yang berbeda. Setiap *input node* terhubung ke semua *hidden layer*. Setiap *node*pada *hidden layer* terhubung dengan *output layer* yang dihubungkan oleh bobot *output*. Semua *hidden node* terhubung ke satu *output*.

Metode ELM mempunyai model matematis yang berbeda dari jaringan saraf tiruan *feedforward*. Model matematis ELM lebih sederhana dan efektif. Berikut langkah – langkah algoritma ELM.

1. Mencari Hasil Keluaran *Hidden Layer*

   Pada *input layer* akan diterima data sebanyak N yang akan diproses oleh *hidden layer*. Pada proses ini, data sebanyak N berukuran data$x$fitur dikalikan dengan bobot awal dan hasilnya dijumlahkan dengan bias sehingga diperoleh matriks keluaran *hidden layer* dari Persamaan 3. Pada ELM bobot *input* dan *hidden bias* ditentukan secara acak. Untuk mendapatkan bobot awal dan bias berupa matriks diperoleh dari Persamaan 1 dan 2.

   $$\boldsymbol{w = h * N} \qquad \textbf{(1)}$$

   Keterangan:

   w = ukuran bobot awal
   h = jumlah *hidden node*
   N = jumlah *input node*

   Hasil dari Persamaan 1 akan diperoleh matriks bobot awal dengan ordo hxN dan hasil Persamaan 2 diperoleh matriks bias dengan ordo hx1 sebagai berikut.

   $$\boldsymbol{b = h * 1} \qquad \textbf{(2)}$$

   Keterangan:

b = ukuran bias
h = jumlah *hidden node*

$$\boldsymbol{H} = (\boldsymbol{N} * \boldsymbol{w^T} + \boldsymbol{b}) \quad \textbf{(3)}$$

Keterangan:
H = matriks keluaran *hidden layer*
N = jumlah *input node*
w^T = matriks transpose bobot awal
b = ukuran bias awal

Hasil dari Persamaan 3 diperoleh matriks keluaran *hidden layer* dengan ordo $banyak\ data\ x\ hidden\ node$.

2. Aktivasi Hasil Keluaran *Hidden Layer*

Fungsi aktivasi merupakan fungsi yang mentransformasikan nilai penjumlahan menjadi sebuah nilai yang dapat diproses lebih lanjut. Fungsi inisangat penting ketika melakukan tahap perhitungan output. Fungsi aktivasi digunakan untuk memetakan nilai dari matriks keluaran *hidden layer*. Fungsi aktivasi yang digunakan yaitu *Sigmoid Biner*.

Fungsi sigmoid biner memiliki rentang nilai dari 0 sampai dengan 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan yang membutuhkan nilai *output* pada interval 0 sampai 1. Namun, fungsi ini juga bisa digunakan oleh jaringan yang nilai output-nya 0 atau 1. Fungsi sigmoid biner sangat baikuntuk menyelesaikan permasalahan kompleks dan bersifat non-linier. Rumus fungsi aktivasi Sigmoid Biner ditunjukkan pada Persamaan 4:

$$\boldsymbol{G(x)} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e^{-x}}} \quad \textbf{(4)}$$

Keterangan:
$G(x)$ = fungsi aktivasi sigmoid biner
e^{-x} = eksponensial pangkat minus data ke-x

3. Mencari Matriks *Moore-Penrose Pseudo Inverse*

Penggunaan umum dari *pseudoinverse* adalah untuk menghitung solusi untuk sistem persamaan linear yang tidak memiliki solusi yang unik. Penggunaan lain adalah untuk menemukan nilai minimal untuk sebuah sistem persamaan linear dengan beberapa solusi. Persamaan untuk mencari matriks *moore-penrose* ditunjukkan pada Persamaan 5.

$$\boldsymbol{H^+} = (\boldsymbol{H^T H})^{-1}\boldsymbol{H^T} \quad \textbf{(5)}$$

Keterangan:

H^+ = *moore-penrose pseudo inverse*
H^T = matriks *H* transpose
H = matriks *H* keluaran *hidden layer*

4. Hitung Bobot *Output*
   Bobot *output* menghubungkan *hidden layer* dengan *output layer* untuk menghitung hasil keluaran. Persamaan untuk memperoleh bobot *output* ditunjukkan pada Persamaan 6.

$$\boldsymbol{\beta = H^{+}T} \quad \textbf{(6)}$$

   Keterangan:
   β = bobot *output*
   H^{+} = matriks *moore-penrose pseudo inverse*
   T = matriks target

5. Hitung *Output*

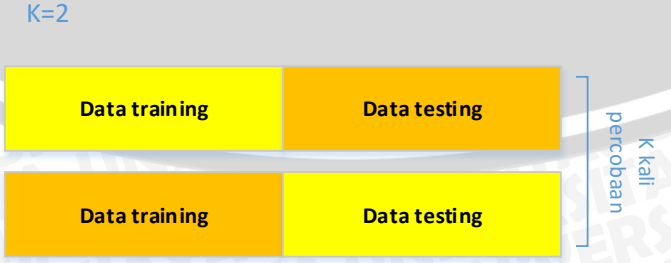
   Hasil keluaran ELM diperoleh dari proses perhitungan dari *hidden layer* dengan bobot *output* yang menghubungkan keduanya. Persamaan menghitung keluaran ditunjukkan pada Persamaan 7.

$$\boldsymbol{O = H\beta} \quad \textbf{(7)}$$

   Keterangan:
   O = *output*
   H = matriks keluaran *hidden layer*
   β = bobot *output*

2.5.1 *K-Fold Cross Validation* (KCV)

Pada penelitian ini banyaknya *datatraining* dan *data testing* digunakan metode K-*Fold Cross Validation* (KCV). Metode KCV membagi *data set* sebanyak varibel nilai k dan dilakukan iterasi sebayak nilai k. Pada penelitian yang dilakukan Anguita dkk (2009) yaitu semua data digunakan sebagai *data training* dan salah satu sebagiannya sebagai *data testing* secara bergantian. Pada proses pembagian data terdapat sisa bagi data, kondisi ini data tersebut akan diletakkan pada kelompok data terakhir. Keunggulan metode ini yaitu bagaimana membagi *data set* yang belum pasti sehingga menghasilkan pembagian dengan optimal. Selain itu, dengan KCV kita bisa memilih seberapa besar pembagian *data set* dan banyak percobaan dari nilai k sesuai kebutuhan pengguna. Berikut pada Gambar 2.9 akan ditunjukkan K-*Fold Cross Validation*.



Gambar 2.6 Metode *K-Fold Cross Validation*

1. Proses *Training*

   Pada penelitian ini proses *training* digunakan untuk menghitung keluaran berupa bobot baru yang akan di proses selanjutnya pada proses *testing* menggunakan *data training*. Proses perhitungan *training* menggunakan Persamaan 6 yaitu operasi matriks sehingga menghasilkan matriks bobot *output*.

2. Proses *Testing*

   Pada proses *training* dilakukan proses perhitungan menghasilkan keluaran berupa prediksi diagnosa kelas dari penentuan kualitas air sungai menggunakan *data testing* dan matriks bobot hasil perhitungan proses *training*. Proses perhitungan menggunakan Persamaan 7 menghasilkan keluaran yang diperoleh akurasi.

2.5.2 Nilai Evaluasi

Evaluasi dilakukan untuk mencari nilai akurasi yang paling baik. Pengukuran tingkat akurasi dilakukan untuk menentukan efisiensi dan ketepatan dari sebuah metode.

Berikut ini merupakan Persamaan 8 untuk menghitung nilai akurasi dari algoritma ELM.

$$\boldsymbol{Akurasi} = \frac{\boldsymbol{N-m}}{\boldsymbol{N}}\% \qquad \textbf{(8)}$$

Keterangan:

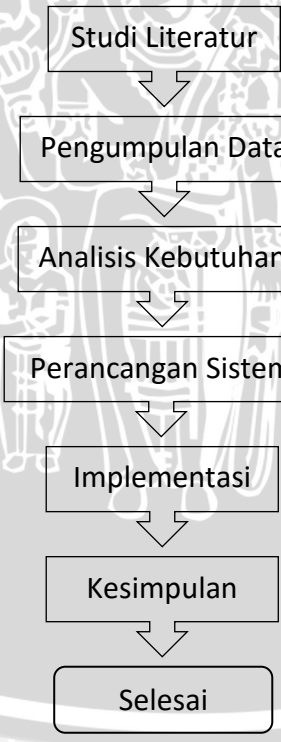
N = Jumlah data keseluruhan

m = data yang tidak sama

BAB 3 METODOLOGI

Metodologi merupakan kumpulan cara (metode) yang lebih spesifik dalam penyelesaian masalah. Metodologi penelitian dapat dipahami sebagai sebuah ilmu untuk mempelajari bagaimana sebuah penelitian dilakukan secara sistematik. Dalam ilmu ini kita mempelajari berbagai langkah yang umumnya digunakan oleh peneliti dalam melakukan proses dari awal penelitian hingga diperoleh hasil. Oleh karena itu di dalam pembahasan metodologi penelitian, yang dibicarakan tidak hanya metode, teknik, atau langkah-langkah yang digunakan dalam sebuah penelitian tetapi juga logika di balik metode, teknik, atau langkah-langkah tersebut sesuai dengan konteks penelitiannya masing-masing. Dalam hal ini perlu dijelaskan mengapa sebuah metode atau teknik dipilih.

Metodologi penelitian merupakan cara ilmiah untuk memperoleh informasi menggunakan berbagai macam prosedur seperti analisis dan teori yang digunakan oleh peneliti untuk suatu disiplin ilmu. Penelitian kali ini mengenai "PENENTUAN KUALITAS AIR SUNGAI DENGAN MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*". Dibawah ini Gambar 3.1 merupakan tahapan dari penelitian yang dilakukan:



Gambar 3.1 Tahapan Penelitian

3.1 Studi Literatur

Studi literatur dalam sebuah penelitian digunakan untuk mendapatkan gambaran tentang apa yang sudah dikerjakan. Dalam penelitian ini, studi

literatur digunakan untuk mendapatkan sumber referensi yang digunakan dalam penulisan teori, pengembangan aplikasi ini serta pengujian implementasi. Landasan teori yang dibutuhkan guna menunjang penelitian ini meliputi:

i. Air Sungai
ii. Metode STORET
iii. Jaringan Saraf Tiruan
iv. Extreme Learning Machine

3.2 Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data parameter pengukuran kualitas air sungai. Nilai dari parameter tersebut dihitung menggunakan metode STORET sehingga diperoleh hasil diagnosa kualitas air sungai. Proses penentuan kualitas dilakukan oleh pakar yaitu Dosen Teknik Pengairan, Fakultas Teknik, Universitas Brawijaya.

3.3 Algoritma yang Digunakan

Proses penentuan air sungai menggunakan algoritma *Extreme Learning Machine* (ELM), selanjutnya hasil dari perhitungan akan dicari nilai akurasi yang paling baik. Algoritma ELM diterjemahkan kedalam kode program menggunakan bahasa pemrograman C#. Bahasa C# sangat fleksibel, penggunaan memori baik dan efisien, serta banyak fitur yang disediakan sehingga mudah melakukan *customization* dalam tampilan program. Penggunaan algoritma ELM pada penelitian ini difokuskan pada *learning speed* dan *input* bobot (w) secara *random* tanpa manual satu persatu sehingga waktu komputasi semakin cepat dan proses komputasi tidak menghabiskan slot memori begitu besar.

3.4 Analisis Kebutuhan

Analisis kebutuhan merupakan langkah awal dalam menentukan apa saja yang dibutuhkan oleh sistem serta implementasi seperti apa yang akan dihasilkan. Analisis kebutuhan yang diperlukan dalam penelitian ini meliputi kebutuhan sistem yang merupakan hal yang paling diperlukan oleh sistem untuk menentukan kondisi yang harus dipenuhi untuk penelitian ini.

Agar kebutuhan sesuai dengan maksud dan tujuan pembuatannya, hal tersebut harus berdasarkan dengan hasil analisis yang telah dilaksanakan berdasarkan kodisi, sehingga hasil analisis yang telah dilaksanakan harus dilakukan pengujian agar diperoleh hasil yang sesuai tujuan.

Kebutuhan aplikasi yang diperlukan terdiri dari:

1. Kebutuhan Hardware, meliputi:

   Notebook dengan spesifikasi Processor Intel(R) Core(TM) i3-2370M CPU @2,40GHz, RAM 4GB, VGA NVIDIA 2GB, System 64-bit OS.

2. Kebutuhan Software, meliputi:
   - Microsoft Windows 8/10
   - Microsoft Excel 2013
   - Visual Studio
3. Data yang dibutuhkan yaitu TSS, BOD, COD, DO, pH, Fenol, Minyak dan Lemak.

3.5 Perancangan Sistem

Perancangan sistem dibangun berdasarkan hasil pengambilan data dan analisis kebutuhan yang telah dilakukan. Pada pengembangan sistem ini, perancangan dilakukan untuk mempermudah implementasi, pengujian dan analisis.

3.5.1 Implementasi Sistem

Implementasi ELM akan dilakukan dalam dua proses yaitu manualisasi data pada Microsoft Excel 2013 dan selanjutnya akan diimplementasikan kedalam code program sehingga diperoleh hasil dari keduanya. Bahasa pemrograman yang digunakan dalam implementasi sistem yaitu C#. Hal ini karena bahasa pemrograman C# *fleksibel*, memiliki *memory management* yang baik sehingga tidak menghabiskan banyak ruang memori, dan banyak tersedia tools sehingga sangat mudah untuk melakukan *setting* tampilan sistem yang baik. Selanjutnya, membandingkan kecocokan nilai yang dihasilkan dari manualisasi dengan kode program. Nilai yang dicocokkan meliputi hasil manualisasi, pembagian data *training* dan *testing*, dan hasil perhitungan ELM.

3.6 Pengujian dan Analisis

Pada tahap ini dilakukan pengujian sistem yang telah dibuat dapat menunjukkan bahwa algoritma ELM bisa bekerja maksimal serta tingkat akurasi yang optimal. Selanjutnya melakukan evaluasi sehingga mengetahui hasil yang nantinya dijadikan sebagai kesimpulan dari penelitian ini. Pengujian algoritma yang dilakukan meliputi pengujian nilai *k* dan pengujian jumlah *hidden node*.

BAB 4 PERANCANGAN SISTEM

Bab ini menjelaskan identifikasi permasalahan, siklus algoritma *Extreme Learning Machine* (ELM), siklus penyelesaian penentuan kualitas air sungai di Waduk Sutami menggunakan algoritma *Extreme Learning Machine* (ELM), perhitungan manual, serta perancangan antarmuka.

4.1 Identifikasi Permasalahan

Permasalahan yang akan diselesaikan dengan ELM adalah penentuan kualitas air sungai, dimana data masukan dari pengguna. Hasilnya akan diperoleh nilai akurasi untuk mengetahui kualitas dari hasil perhitungan ELM dari setiap unit masukan. Masukan dari pengguna meliputi data dalam format .xls terdiri dari 7 parameter kualitas air sungai, yaitu residu tersuspensi (TSS), BOD, COD, DO, pH, Fenol, serta Minyak dan Lemak.

Sedangkan, masukan dari sistem sendiri meliputi parameter perhitungan terdiri dari: jumlah *hidden nodes*, bobot (w) dan bias (b). Penentuan bobot awal berdasarkan banyaknya *input* dan *hidden node*. Pada penelitian ini ada 7 fitur dengan 2 *hidden node* sehingga diperoleh ukuran bobot antara *input layer* dan *hidden layer*$(w_{ij})2 \times 7$, bias antara *input layer* dan *hidden layer*(b_i) dengan ukuran 2x1. Selanjutnya, masuk tahap *training*. Pada tahap ini, data akan dimulai proses perhitungan fungsi keluaran *hidden layer* dan diaktivasi, kemudian diperoleh hasil berupa matriks. Selanjutnya, hasil dari perhitungan tersebut akan diperoleh bobot *output*. Setelah didapat bobot *output* akan masuk kedalam proses *testing* untuk menghitung keluaran dari sistem. Hasil keluaran dari unit *output* akan dihitung nilai akurasinya.

Adapun sampel data air sungai yang digunakan pada penelitian sebagai berikut pada Tabel 4.1 (Lihat Lampiran 1):

Tabel 4.1 Data Air Sungai

Fisika	Kimia				Kimia	
TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak
0	15.35	56.8	11.2	7.5	0.775	0.8
0	10	34.8	15.3	8.05	0.0145	4.5
44.85	12.6	35.4	18.55	8.85	0.1185	4.25
72.75	19.75	62.85	10.45	8.1	0.172	0
10	6.2	13.9	10.15	7.55	0.172	1.5
6.2	11.3	35.1	16.925	8.45	0.086	1.425

24.25	6.95	15.6	10.85	7.3	0,067	0.8
26.25	7.9	21	11.75	7	0.196	0
17.4	6.9	16.3	7.95	6.8	0.128	0.9
21.25	6.25	13.4	7.45	6.95	0.0925	0.8
17.75	5.05	20.6	3.5	6.9	0.0885	0.8
25.35	4.1	16.85	6.75	7	0.151	3.5

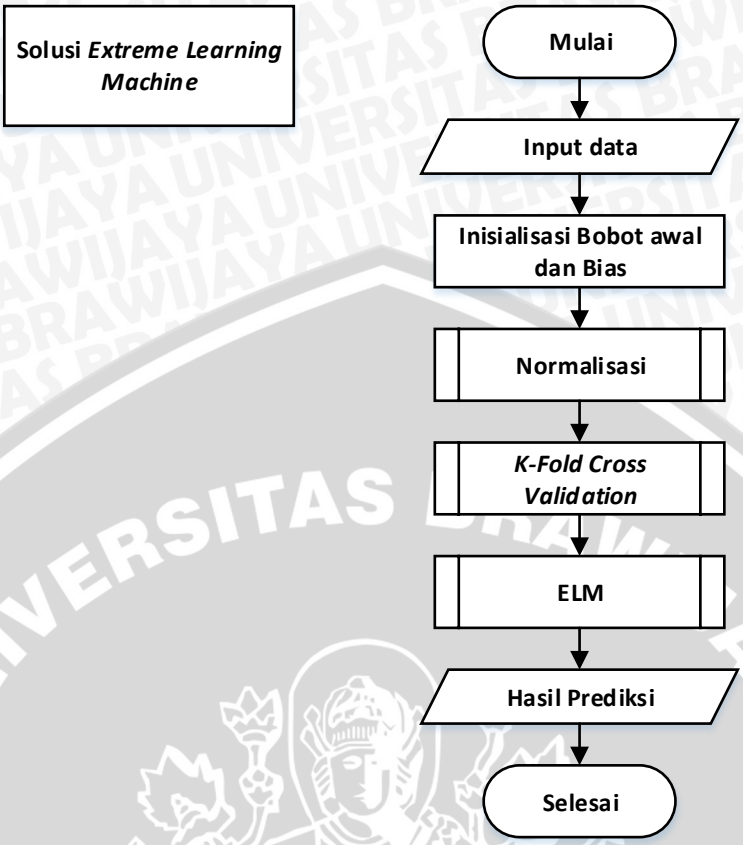
Atribut data terpilih telah dikonsultasikan dengan narasumber terkait variabel – variabelyang digunakan sebagai pertimbangan dalam menentukan kualitas air sungai. Keterangan setiap atribut data pada Tabel 4.1 adalah sebagai berikut

1. TSS : berupa lumpur, tanah liat, logam oksida, sulfida, ganggang (double)
2. BOD : oksigen terlarut untuk penguraian mikroorganisme (double)
3. COD : kandungan oksidatator kuat untuk penguraian (double)
4. DO : kandungan oksigen terlarut baik (double)
5. pH : tingkat keasaman (double)
6. Fenol : kandungan zat kristal tak berwarna yang memiliki bau khas (double)
7. Minyak : cairan organik yang tidak larut/bercampur dalam air (double)

4.2 Siklus Penyelesaian Algoritma *Extreme Learning Machine* (ELM)

Siklus penyelesaian algoritma ELM merupakan urutan penyelesaian masalah menggunakan algoritma ELM secara sekuensial. Pada siklus algoritma yang akan dijelaskan pada sub bab berikutnya,selain proses penentuan kualitas menggunakan algoritma ELM, telah ditambahkan proses normalisasi data menggunakan metode *Min-Max*.

Untuk penentuan data pada proses *training* dan *testing* menggunakan metode *K-Fold Cross Validation*, yang merupakan metode dengan membagi data sebanyak nilai *k*. Data secara bergantian sebagai data *training* dan satu diantaranya sebagai data *testing*. Berikut ini alur penyelesaian menggunakan algoritma ELM pada Gambar 4.1.



Gambar 4.1 Siklus Penyelesaian Menggunakan Algoritma ELM

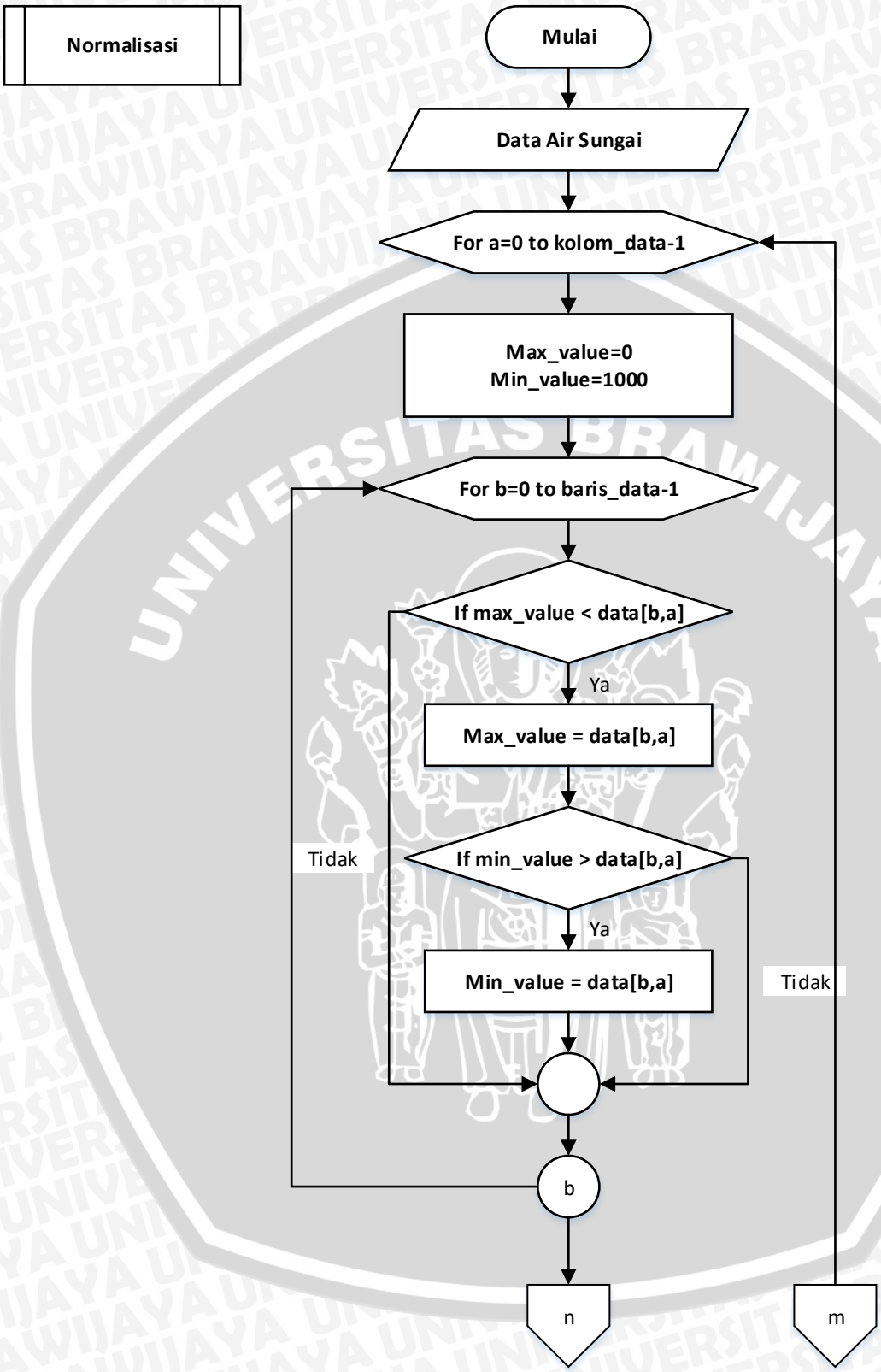
4.2.1 Proses Normalisasi Data

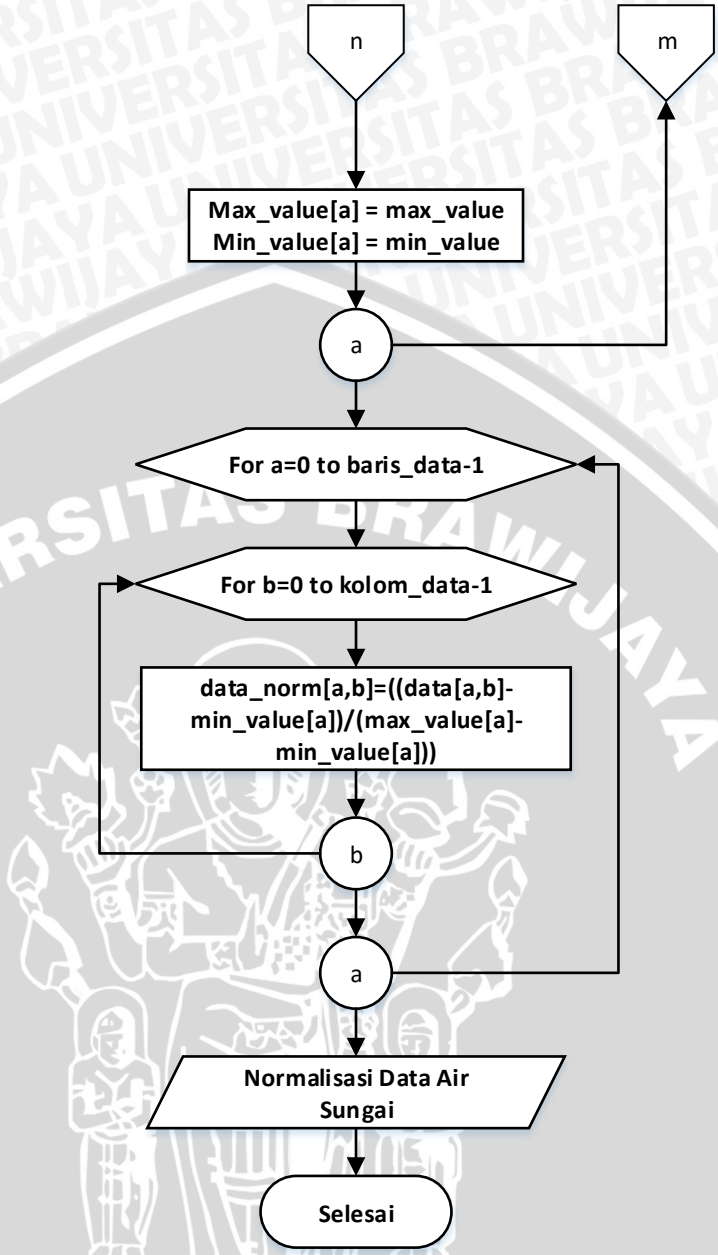
Normalisasi data merupakan proses tranformasi nilai atribut dari suatu data tertentu dalam rentang nilai tertentu. Normalisasi bertujuan untuk mengurangi kesalahan dalam proses *data mining*. Pada penelitian ini, teknik normalisasi yang digunakan adalah *Min-Max*. Metode *Min-Max* merupakan metode normalisasi sederhana yang melakukan proses transformasi linier terhadap data awal. Keunggulan *Min-Max* perbandingan nilai antara sebelum dan sesudah proses normalisasi seimbang. Fungsi normalisasi dapat dituliskan pada Persamaan 9 sebagai berikut (Santosh dkk, 2014) dan alur proses normalisasi menggunakan *Min-Max* digambarkan *flowchart* berikut pada Gambar 4.1.

$$\boldsymbol{x} = \frac{(\boldsymbol{data_{awal}} - \boldsymbol{data_{min}})}{(\boldsymbol{data_{max}} - \boldsymbol{data_{min}})} \quad \textbf{(9)}$$

Keterangan:
x = data normalisasi
$data_{awal}$ = data inputan
$data_{min}$ = nilai minimal dari data inputan
$data_{max}$ = nilai maksimal dari data inputan

Nilai yang diperoleh dari Persamaan 10 berupa nilai baru berada pada rentang 0 sampai 1.


Normalisasi
Mulai
Data Air Sungai
For a=0 to kolom_data-1
Max_value=0
Min_value=1000
For b=0 to baris_data-1
If max_value < data[b,a]
Ya
Max_value = data[b,a]
Tidak
If min_value > data[b,a]
Ya
Min_value = data[b,a]
Tidak
b
n
m



Gambar 4.2 *Flowchart* Proses Normalisasi

Berdasarkan Gambar 4.2 diatas, langkah – langkah proses normalisasi data menggunakan metode *Min-Max* sebagai berikut:

a. Masukan berupa data air sungai dalam format .xls akan diterima oleh sistem.
b. Sistem akan melakukan perhitungan mencari nilai terkecil dan terbesar dari setiap nilai atribut masukan. Pada penelitian ini terdapat 7 *input* yang akan dihitung nilai *minimum* dan *maximum*.
c. Melakukan proses perhitungan normalisasi menggunakan Persamaan 9.
d. *Output* dari proses normalisasi berupa data dengan rentang 0 sampai 1.

Langkah – langkah perhitungan normalisasi berdasarkan Persamaan 9 menggunakan metode *Min-Max* sebagai berikut:

Langkah 1: mencari nilai *maximum* dan *minimum* dari setiap nilai atribut. Berikut nilai *maximum* dan *minimum* dari setiap inputan data air sungai ditunjukkan pada Tabel 4.2.

Tabel 4.2 Nilai *Maximum* dan *Minimum*Parameter Air Sungai

	TSS	BOD	COD	DO	pH	Fenol	Minyak & Lemak
Max	229.8	21.3	97.5	42.25	9	0.3825	6.85
Min	0	1.925	6.408	3.5	6.55	0	0

Langkah 2: hitung nilai normalisasi dari masing – masing nilai data air sungai menggunakan metode *Min-Max*. Berikut contoh menghitung nilai normalisasi data ke-1 yaitu data bulan Januari tahun 2005.

$$x_{1,1} = \frac{0 - 0}{229.8 - 0} = 0$$

$$x_{1,2} = \frac{15.35 - 1.925}{21.3 - 1.925} = 0.692903$$

$$x_{1,3} = \frac{56.8 - 6.408}{97.5 - 6.408} = 0.555333$$

$$x_{1,4} = \frac{11.2 - 3.5}{42.5 - 3.5} = 0.19871$$

$$x_{1,5} = \frac{7.5 - 6.55}{9 - 6.55} = 0.387755$$

$$x_{1,6} = \frac{0.0775 - 0}{0.3825 - 0} = 0.202614$$

$$x_{1,7} = \frac{0.8 - 0}{6.85 - 0} = 0.116788$$

Tabel 4.3 Hasil Normalisasi Data Air Sungai

Data ke-	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$	$x_{1,6}$	$x_{1,7}$
1	0	0.692903	0.555333	0.19871	0.387755	0.202614	0.116788
2	0	0.416774	0.312887	0.304516	0.612245	0.037908	0.656934

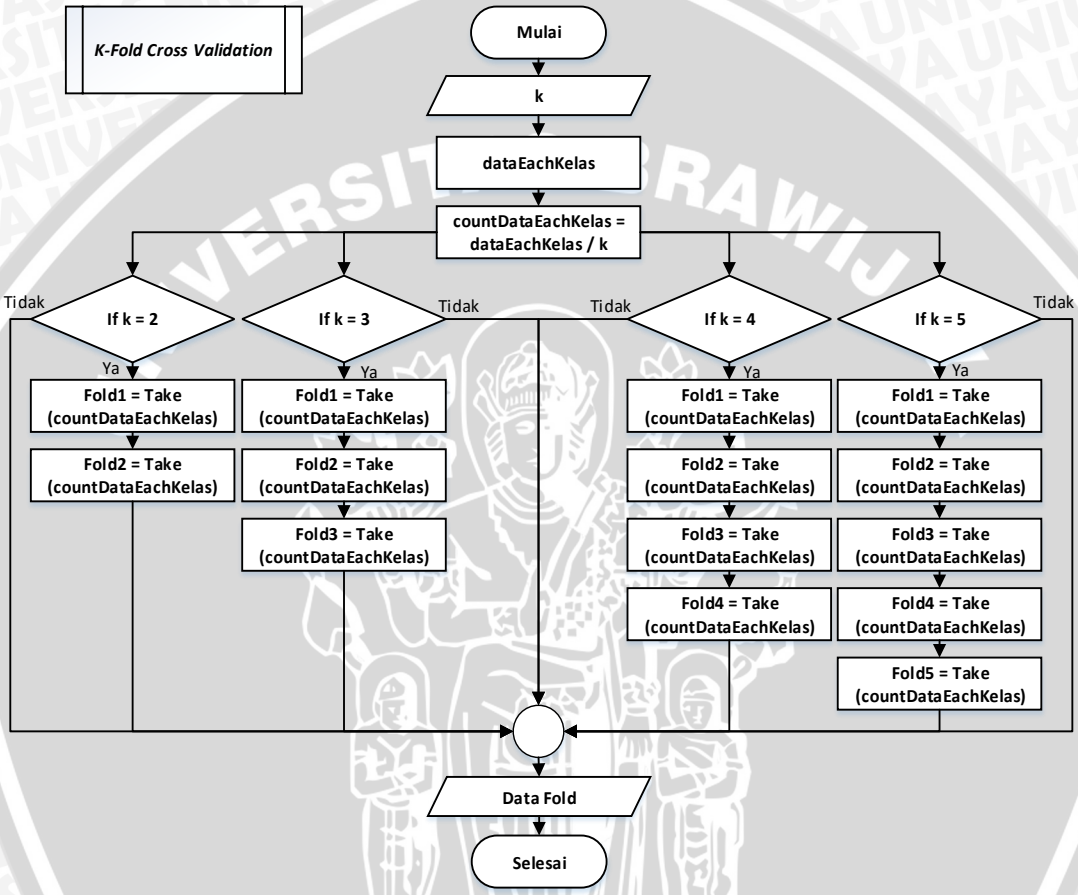
4.2.2 Proses Pengambilan *Data Training* dan *Data Testing* Menggunakan K-*Fold Cross Validation*

Metode KCV membagi *data set* sebanyak varibel nilai k dan dilakukan iterasi sebayak nilai k. Pada penelitian yang dilakukan Anguita dkk (2009) yaitu semua data digunakan sebagai *data training* dan salah satu sebagiannya sebagai *data testing* secara bergantian. Pengambilan data diletakkan pada masing – masing *fold*dimana data yang sudah diambil tidak boleh diletakkan pada *fold* yang lainnya. Pada manualisasi ini jumlah data yang digunakan sebanyak 60 data dengan 3 Kelas. Pengambilan untuk masing – masing dari setiap Kelas dapat

menggunakan Persamaan 10. Proses pengambilan data ditunjukkan pada Gambar 4.3.

$$\boldsymbol{X = \frac{100}{k}\%} \quad \textbf{(10)}$$

Keterangan:
x = persentase pembagian
k = jumlah *fold*



Gambar 4.3 *Flowchart* Proses Pengambilan Data Menggunakan KCV

Berdasarkan Gambar 4.3 langkah – langkah pengambilan data menggunakan K-*Fold Cross Validation* sebagai berikut:

a. Inisialisasi nilai k. Pada contoh manualisasi menggunakan nilai $k = 3$ yang berarti terdapat 3 *fold* yaitu *fold*-1, *fold*-2, dan *fold*-3.
b. Untuk pengambilan data pada masing – masing Kelas menggunakan Persamaan 10 dan diletakkan pada setiap *fold*. Data yang telah diambil dan diletakkan pada *fold* tertentu tidak boleh diambil dan diletakkan pada *fold* yang lain. Hal ini menunjukkan bahwa data pada setiap *fold* tidak boleh sama seperti pada Gambar 4.4.

Langkah – langkah pengambilan data berdasarkan Persamaan 10 menggunakan metode K-*Fold Cross Validation* sebagai berikut:

Langkah 1: membagi data sebanyak nilai $k = 3$. Data dibagi sebanyak 3 yang terdiri dari 1 data *testing* dan 2 data *training* secara bergantian.

Data ke-	Fold 1	Fold 2	Fold 3
1			
2			

Gambar 4.4 Pembagian Data Pada Setiap *Fold*

Langkah 2: hitung banyaknya data dari setiap Kelas yang diambil menggunakan Persamaan 10. Untuk sisa data akan diletakkan pada *fold* terakhir.

$$x = \frac{100}{3}\% = 33{,}33\%$$

Banyaknya data dari masing – masing Kelas yang diambil sebanyak 33,33% pada Tabel 4.4.

Tabel 4.4 Pengambilan Data Pada Setiap Kelas

Jumlah	Kelas 1	Kelas 2	Kelas 3
Data	3	44	13
Pengambilan setiap *Fold*	1	14	4

Contoh data hasil pengambilan yang diletakkan pada *fold*-1 sebagai *data training* pada Tabel 4.5 dan *fold*-3 pada Tabel 4.6 sebagai *data testing* dapat dilihat (Lihat Lampiran 6).

Tabel 4.5 Pengambilan Data *Training Fold* 1 dan 2

No	TSS	BOD	COD	DO	pH	Fenol	Lemak dan Minyak	Kelas
13	0.152306	0.029677	0.052809063	0.139354839	0.346939	0.491503	0.1167883	1
1	0	0.692903	0.555332702	0.198709677	0.387755	0.202614	0.1167883	2
...	...	...	...	...	...	...	...	...
37	0.291123	0.174194	0.047023	0.054194	0.183673	0.227451	0.59854	3

Tabel 4.6 Pengambilan Data *Testing Fold* 3

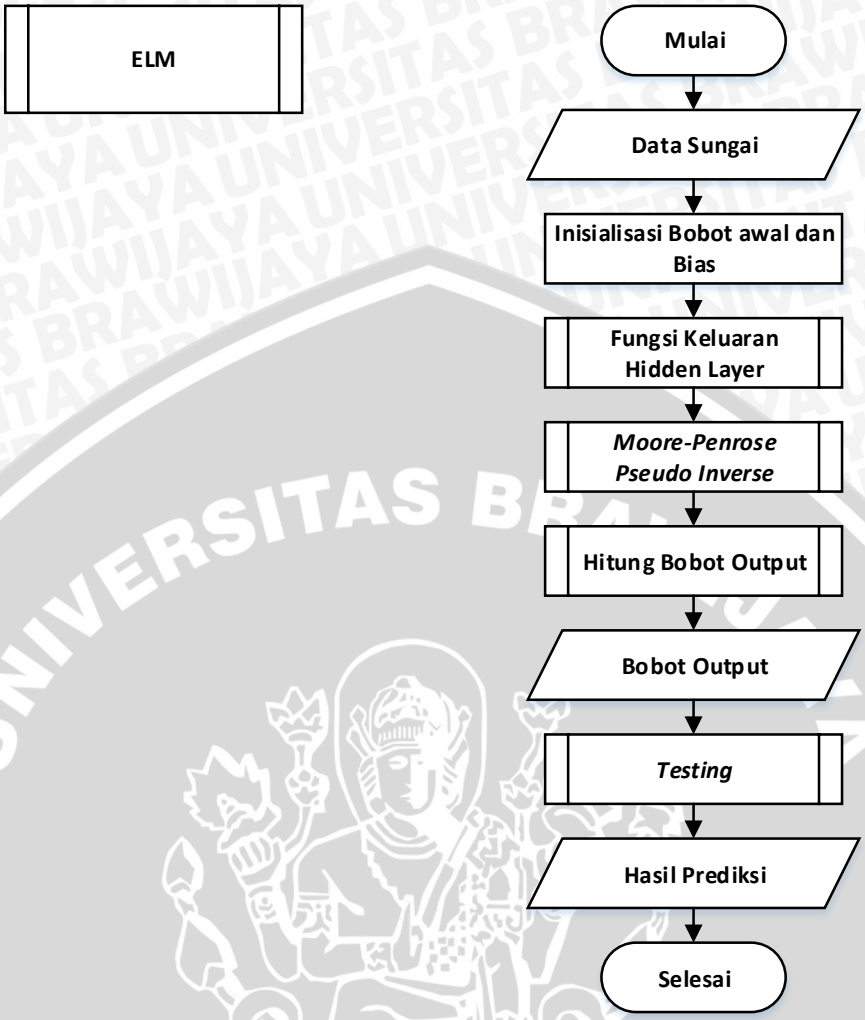
No	TSS	BOD	COD	DO	pH	Fenol	Lemak dan Minyak	Kelas
55	0.0876	0	0	0.1122	0.5306	0	0.41605	1

	85			58	12		8	
39	0.367711	0.029677	0.020299	0.175484	0.693878	0.151634	0.729927	2
…	…	…	…	…	…	…	…	…
58	0.184073	0.094194	0.036692	1	0.22449	0	0.693431	3

4.2.3 Proses Penentuan Kualitas Air Sungai Menggunakan Algoritma *Extreme Learning Machine*

Algoritma ELM digunakan untuk memperoleh hasil akhir kualitas air sungai berdasarkan perhitungan dengan bobot awal dan bias yang memenuhi. Penentuan bobot akhir berdasarkan hasil perhitungan menggunakan keluaran dari *hidden layer* dengan matriks *moore-penrose pseudo inverse*. Penentuan bobot awal dan bias menggunakan Persamaan 1 dan 2. Selanjutnya, perhitungan keluaran dari *hidden layer* mengunakan Persamaan 3. Hasil keluaran tersebut akan dihitung menggunakan fungsi aktifasi *sigmoid biner* untuk pemetaan nilai keluaran dari setiap *hidden layer* ke dalam rentang nilai 0 sampai dengan 1 menggunakan Persamaan 4.

Perhitungan bobot akhir ini digunakan dalam perhitungan hasil keluaran dari algoritma ini. Perhitungan nilai bobot akhir menggunakan Persamaan 6 sedangkan menghitung keluaran menggunakan Persamaan 7. Untuk memperoleh hasil akhir variabel yang digunakan dalam proses perhitungan yaitu matriks keluaran *hidden layer* dan bobot *output* dari hasil perkalian *moore-penrose pseudo inverse* dengan matiks target. Setelah diperoleh hasil keluaran dilakukan proses evaluasi untuk memperoleh nilai akurasi yang baik. Hasil evaluasi tersebut akan diambil nilai akurasi yang paling tinggi diantaranya. Perhitungan tingkat akurasi menggunakan Persamaan 8. Berikut alur proses perhitungan ELM pada Gambar 4.5.



Gambar 4.5 *Flowchart* ELM

Berdasarkan Gambar 4.5 langkah – langkah proses perhitungan ELM sebagai berikut:

a. Inisialisasi bobot awal dan bias menggunakan Persamaan 1 dan 2. Bobot awal dan bias diambil secara acak dengan rentang nilai 0 sampai 1.
b. Menghitung keluaran dari *hidden layer* menggunakan Persamaan 3 yang hasilnya berupa matriks dan menggunakan fungsi aktivasi untuk memetakan hasil keluaran tersebut. Fungsi aktifasi yang digunakan pada proses manualisasi yaitu *Sigmoid Biner* pada Persamaan 5.
c. Menghitung matriks *moore-penrose pseudo inverse* menggunakan Persamaan 6.
d. Menghitung hasil keluaran dari proses *training* menggunakan Persamaan 7. Hasil keluaran berupa bobot *output* digunakan pada proses *testing*.
e. Melakukan proses *testing* untuk menghitung keluaran dari sistem.

Langkah – langkah proses inisialisasi bobot awal dan bias menggunakan Persamaan 1 dan 2 sebagai berikut:

Langkah 1: jumlah *hidden node* yang digunakan dalam manualisasi sebanyak 2 dan jumlah *input layer* sebanyak 7. Maka, menggunakan Persamaan 1 dapat diperoleh bobot awal secara acak.

ordo $w = 2x7$

Bobot awal dalam bentuk matriks dengan ordo 2x7 sebanyak 14 nilai. Berikut contoh bobot awal yang diambil secara acak pada Tabel 4.7.

Tabel 4.7 Nilai Bobot Awal

w	1	2	3	4	5	6	7
1	0.554463	0.512528	0.524558	0.756425	0.513373	0.768145	0.57825725
2	0.159467	0.255763	0.160995	0.629833	0.811404	0.916429	0.551484879

Matriks pada Tabel 4.7 merupakan hasil dari perkalian jumlah *hidden node* sebanyak 2 dengan jumlah *input node* sebanyak 7 menghasilkan matriks bobot awal ordo 2x7. Nilai dari matriks bobot awal diatas diambil secara acak dengan rentang nilai 0 sampai 1.

Langkah 2:hitung bias menggunakan Persamaan 2 dengan jumlah *hidden node* sebanyak 2.

ordo $b = 2x1$

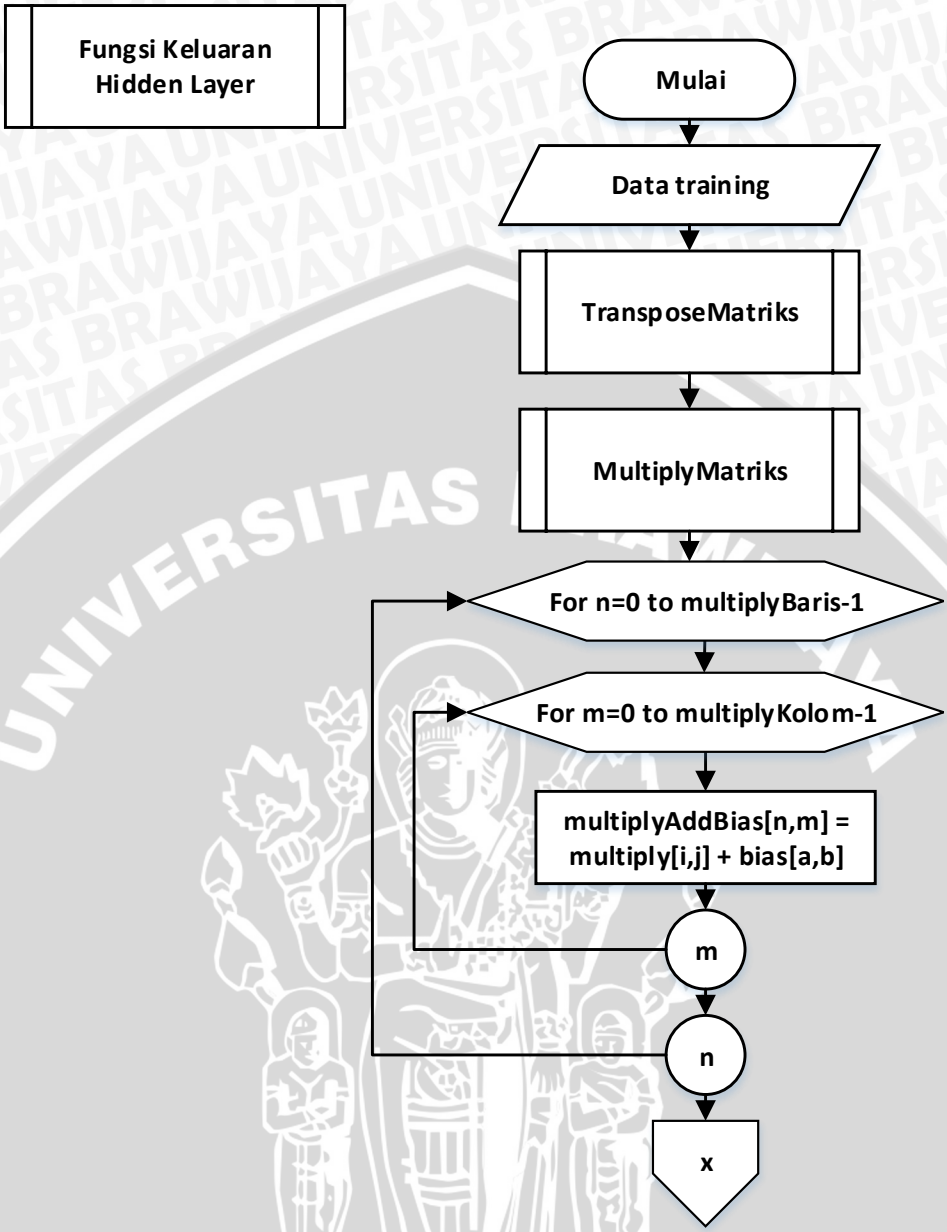
Bias dalam bentuk matriks ordo 2x1 sebanyak 2 nilai. Berikut contoh bias awal yang diambil secara acak pada Tabel 4.8.

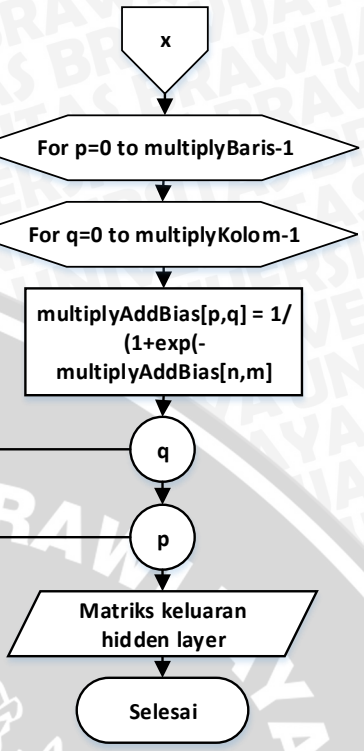
Tabel 4.8 Nilai Bias

b	1
1	0.352154
2	0.798536

Matriks pada Tabel 4.8 merupakan hasil perkalian jumlah *hidden node* kali 1 menghasilkan matriks bias ordo 2x1. Nilai dari matriks bias diambil secara acak dengan rentang nilai 0 sampai 1.

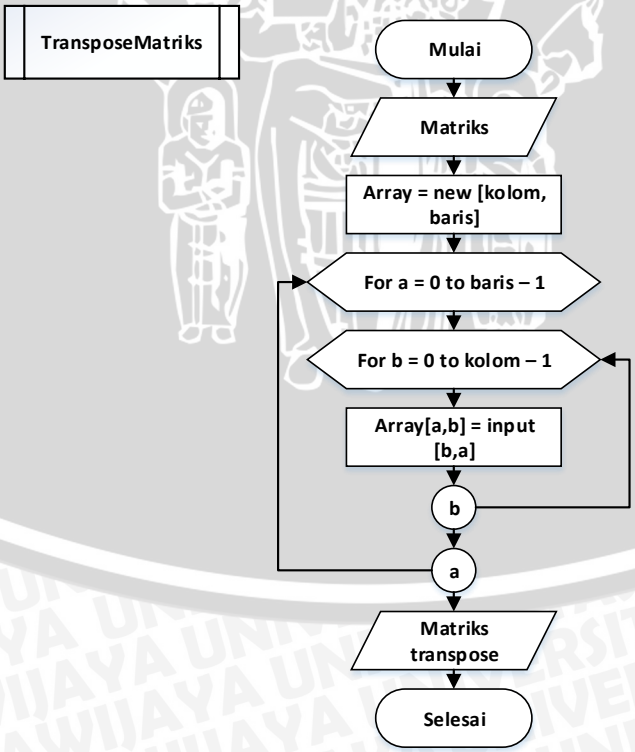
Langkah 3: menghitung keluaran setiap *hidden layer* dan hasil perhitungannya akan dipetakan menggunakan fungsi aktifasi *Sigmoid Biner*. Berikut contoh menghitung matriks keluaran *hidden layer* menggunakan data 1 dari data *training* pada Tabel 4.8 dan perhitungan menggunakan Persamaan 3. Langkah – langkah proses mencari matriks keluaran *hidden layer* ditunjukkan pada Gambar 4.6. Berikut ini contoh perhitungan dan hasilnya.


Fungsi Keluaran Hidden Layer
Mulai
Data training
TransposeMatriks
MultiplyMatriks
For n=0 to multiplyBaris-1
For m=0 to multiplyKolom-1
multiplyAddBias[n,m] = multiply[i,j] + bias[a,b]
m
n
x



Gambar 4.6 *Flowchart* Fungsi Keluaran *Hidden Layer*

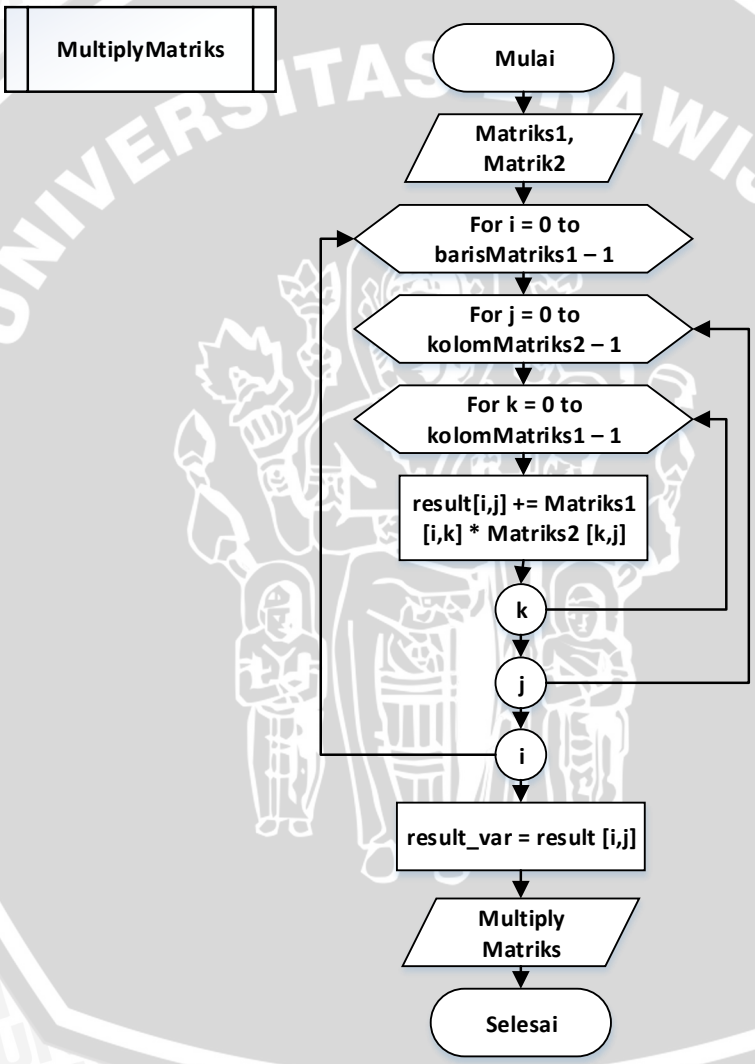
Berdasarkan Persamaan 3, proses perkalian dua matriks yaitu data *training* dengan bobot awal yang ditranspose ditunjukkan pada Gambar 4.7 dan 4.8.



Gambar 4.7 *Flowchart* Fungsi Transpose Matriks

Berdasarkan Gambar 4.7 diatas, *input* yang diproses oleh fungsi matriks transpose berupa bobot awal dan hasil keluaran dari *hidden layer*. Langkah – langkah proses transpose matriks dari baris berubah menjadi kolom dan sebaliknya sebagai berikut.

a. Inisialisasi nilai masukan yang terdiri dari baris dan kolom.
b. Proses transpose matriks mengubah baris dan kolom awal suatu variabel memiliki ordo baris$x$kolom menjadi kolom dan baris baru yang disimpan pada variabel baru dengan ordo kolom$x$baris.
c. Variabel baru tersebut merupakan hasil transpose matriks berupa matriks baru.



Gambar 4.8 *Flowchart* Fungsi Perkalian Matriks

Berdasarkan Gambar 4.8 diatas, *input* yang diproses oleh fungsi perkalian matriks berupa *input* data dengan bobot awal dan proses pada perhitungan mencari matriks *moore-penrose* dengan syarat jumlah baris pada matriks pengali harus sama jumlahnya dengan kolom matriks yang dikalikan . Langkah – langkah proses perkalian matriks sebagai berikut.

a. Inisialisasi baris dan kolom matriks A dan kolom dari matriks B.
b. Proses perkalian matriks dimulai dengan baris matriks A dikalikan kolom matriks B yang dilakukan sebanyak jumlah baris matriks A.
c. Hasil perkalian akan disimpan kedalam matriks baru.

Berdasarkan Persamaan 3, contoh perhitungan manual fungsi keluaran *hidden layer* pada baris 1 kolom 1 sebagai berikut.

$$H_{(1,1)} = [(0.146214 * 0.554463) + (0.029677 * 0.512528) + (0.025809 * 0.524558) + (0.068387 * 0.756425) + (0.408263 * 0.513373) + (0.101961 * 0.768145) + (0.056934 * 0.578257)] + 0.352154 = 1.41756$$

Tabel 4.9 merupakan hasil dari perkalian *input* data ordo 38x7 dengan matriks bobot ordo 7x2 menghasilkan matriks keluaran *hidden layer* ordo 38x2 yang setiap kolomnya dijumlahkan dengan kolom yang sama pada bias (Lihat Lampiran 7).

Tabel 4.9 Fungsi Keluaran *Hidden Layer*

H	1	2
1	1.208114	1.723029
2	1.571134	1.755029
...	...	...
38	1.283628	1.618778

Langkah 4: hitung matriks keluaran *hidden layer* menggunakan fungsi aktifasi *Sigmoid Biner* untuk memetakan nilainya. Berikut perhitungan matriks fungsi aktifasi menggunakan Persamaan 4.

$$H(x)_{(1,1)} = \frac{1}{1+e^{-0.834487}} = 0.697303$$

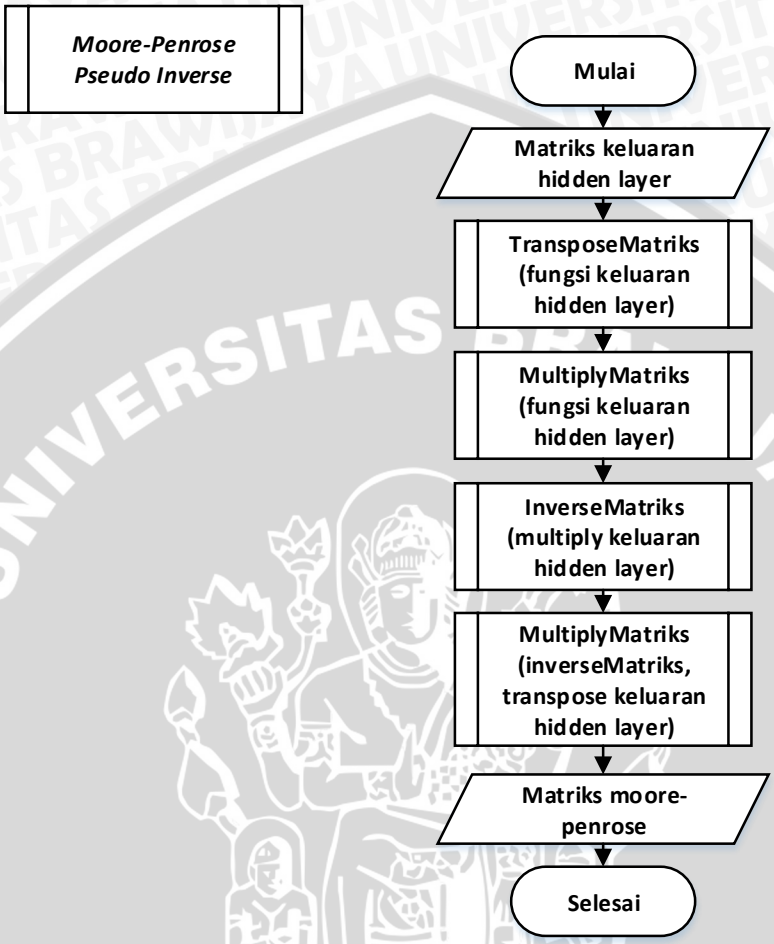
Tabel 4.10 merupakan hasil dari perhitungan fungsi aktifasi *Sigmoid Biner* yang memetakan nilai dari matriks keluaran *hidden layer* (Lihat Lampiran 8). Matriks diatas memiliki nilai yang lebih kecil sehingga memudahkan dalam operasi pada matriks selanjutnya.

Tabel 4.10 Aktifasi Fungsi Keluaran *Hidden Layer*

H(x)	1	2
1	0.76996515	0.848518543
2	0.8279452	0.852586012
...	...	...
38	0.78306679	0.834626532

Langkah 5: Menghitung bobot baru. Perhitungan ini mencari nilai matriks *moore-penrose pseudo inverse*. Bobot baru diperoleh dari hasil perkalian matriks *moore-penrose pseudo inverse* dengan matriks target yang merupakan kelas dari data *training*. Hasil berupa bobot output untuk proses *testing* mencari hasil prediksi

dari penentuan kualitas air sungai. Langkah – langkah proses mencari matriks *moore-penrose* ditunjukkan pada Gambar 4.9. Contoh perhitungan dari manualisasi program untuk mencari matriks *moore-penrose* menggunakan Persamaan 5.

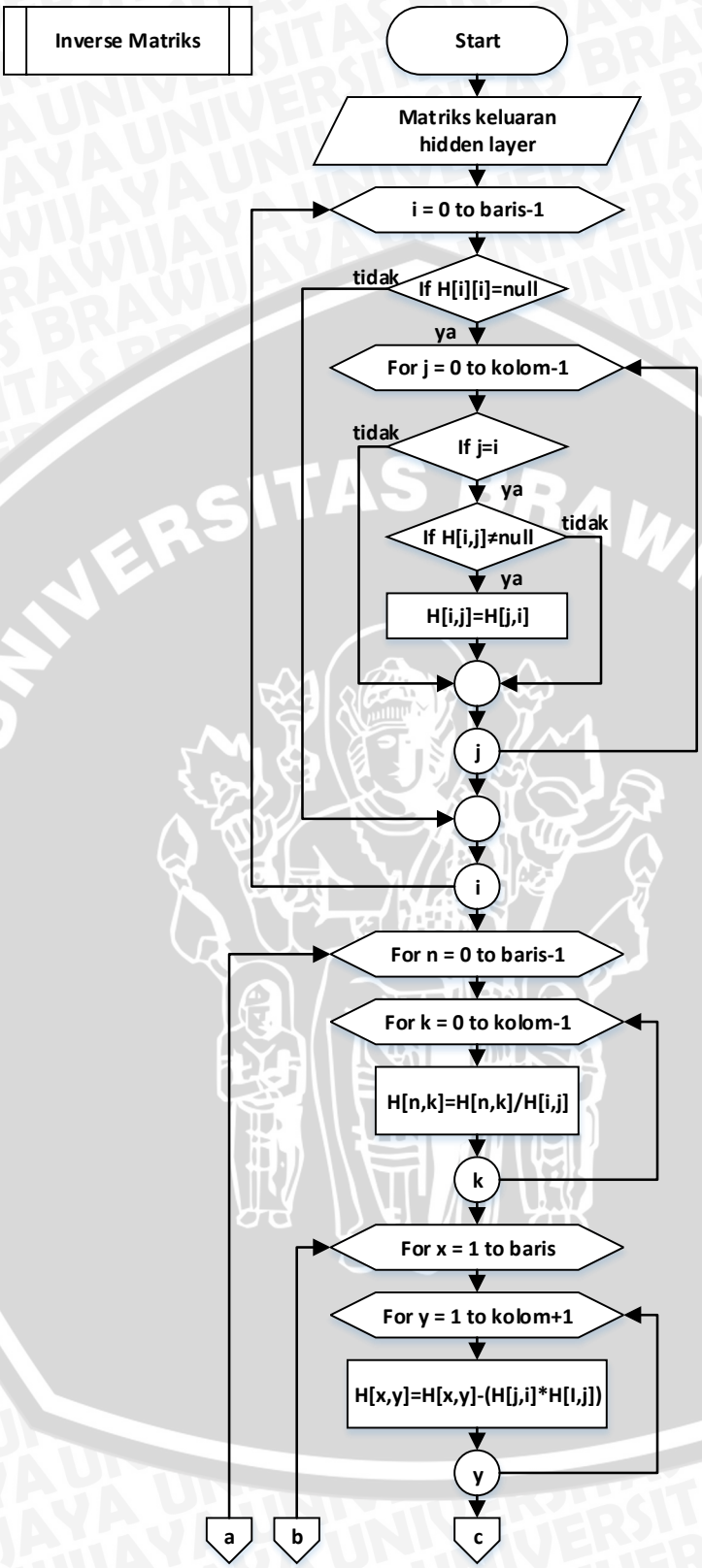


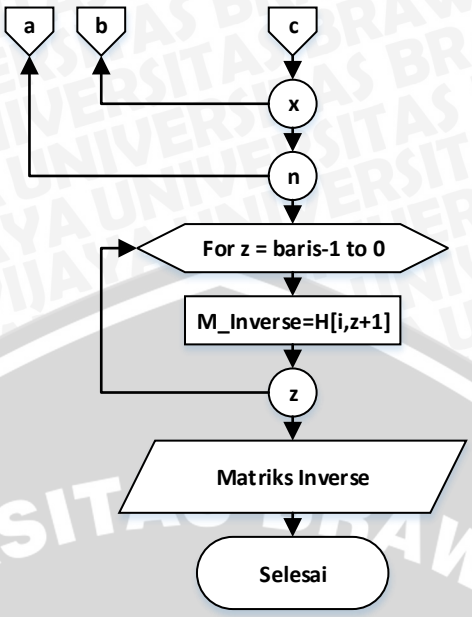
Gambar 4.9 *Flowchart* Matriks *Moore-Penrose Pseudo Inverse*

Berdasarkan Gambar 4.9 diatas langkah – langkah proses mencari matriks *moore-penrose* sebagai berikut.

a. Inisialisasi matriks keluaran *hidden layer*.
b. Mentranspose matriks keluaran *hidden layer* mengubah baris dan kolom menjadi matriks baru dengan ordo kolom$x$baris.
c. Melakukan proses perkalian matriks keluaran *hidden layer* dengan matriks tranposenya.
d. Hasil perkalian akan diinvers kemudian matriks invers tersebut akan dikalikan dengan matriks transpose keluaran *hidden layer* yang akan diperoleh matriks *moore-penrose*.

Proses invers matriks hasil perkalian fungsi keluaran *hidden layer* dengan transposenya ditunjukkan pada Gambar 4.10 sebagai berikut.


Inverse Matriks
Start
Matriks keluaran hidden layer
i = 0 to baris-1
tidak
If H[i][i]=null
ya
For j = 0 to kolom-1
tidak
If j=i
ya
If H[i,j]≠null
tidak
ya
H[i,j]=H[j,i]
j
i
For n = 0 to baris-1
For k = 0 to kolom-1
H[n,k]=H[n,k]/H[i,j]
k
For x = 1 to baris
For y = 1 to kolom+1
H[x,y]=H[x,y]-(H[j,i]*H[I,j])
y
a
b
c



Gambar 4.10 *Flowchart Inverse* Matriks

Berdasarkan Gambar 4.10, langkah – langkah proses mencari matriks inverse sebagai berikut.

a. Inisialisasi matriks identitas sesuai dengan bentuk matriksnya $(A|I)$. Pada perhitungan invers, matriks identitas digunakan dalam operasi baris elementer.

b. Cek kondisi apabila salah satu baris isinya semua nilai 0, maka matriks A tidak memiliki invers karena determinannya 0 dan proses akan langsung berhenti.

c. Proses operasi baris elementer mengubah matriks A menjadi matriks identitas, sehingga matriks identitas (I) menjadi matriks baru.

d. Setelah matriks A berubah menjadi matriks identitas, maka diperoleh matriks baru dari perubahan matriks identitas yang merupakan invers dari matriks A.

Langkah 5.1: Mengalikan matriks kebalikan dengan matriks $H(x)$. Berikut contoh perhitungan manualnya.

$$H(x)^T H(x)_{(1,1)} = (0.697302765 * 0.697302765) + (0.769965147 * 0.769965147) + \cdots + \left(\ldots_{(n)} * \ldots_{(n)}\right) = 23.68653061$$

Hasil dari perhitungan diatas menghasilkan matriks ordo 2x2 ditunjukkan pada Tabel 4.11 dibawah ini.

Tabel 4.11 Hasil Perkalian Matriks H(x) Transpose Dengan Matriks H(x)

$H(x)^T H(x)$	1	2
1	23.68653061	25.3802904
2	25.3802904	27.22694925

Langkah 5.2: Menghitung matriks *inverse* dari hasil perkalian matriks H dengan transposenya ordo 2x2 menggunakan OBE berdasarkan Gambar 4.10. Berikut contoh perhitungan manualisasinya.

Bentuk matriks identitas $(\boldsymbol{I})$: $[\boldsymbol{H}|\boldsymbol{I}] = \left(\begin{bmatrix} \mathbf{23.68653061} & \mathbf{25.3802904} \\ \mathbf{25.3802904} & \mathbf{27.22694925} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}\right)$

Tukar baris 2 dan baris 1 $\left(\begin{bmatrix} 25.3802904 & 27.22694925 \\ 23.68653061 & 25.3802904 \end{bmatrix} \middle| \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right)$

$R_1 \div 25.3802904 \rightarrow R_1 = \left(\begin{bmatrix} 1 & 1.073 \\ 23.68653061 & 25.3802904 \end{bmatrix} \middle| \begin{bmatrix} 0 & 0.039 \\ 1 & 0 \end{bmatrix}\right)$

$R_2 - 23.68653061R_1 \rightarrow R_2 = \left(\begin{bmatrix} 1 & 1.073 \\ 0 & 0.037 \end{bmatrix} \middle| \begin{bmatrix} 0 & 0.039 \\ 1 & 0.933 \end{bmatrix}\right)$

$R_2 \div (-0.037) \rightarrow R_2 = \left(\begin{bmatrix} 1 & 1.073 \\ 0 & 1 \end{bmatrix} \middle| \begin{bmatrix} 0 & 0.039 \\ -27.165 & 25.356 \end{bmatrix}\right)$

$R_1 - (1{,}073)R_2 \rightarrow R_1 = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \middle| \begin{bmatrix} 29.145 & -27.165 \\ -27.165 & 25.356 \end{bmatrix}\right)$

Matriks hasil perkalian matriks H dengan transposenya memiliki ordo 2x2. Berikut contoh hasil manualisasinya pada Tabel 4.8.

Tabel 4.12 Inverse Hasil Perkalian Matriks H(x) Transpose Dengan Matriks H(x)

$(H(x)^T H(x))^{-1}$	1	2
1	29.145	-27.165
2	-27.165	25.356

Langkah 5.3: Menghitung matriks *moore-pensrose pseudo inverse* dengan mengalikan matriks *inverse* dengan matriks $H(x)$. Berikut contoh perhitungan manualisasinya.

$$H^{+}{}_{(1,1)} = (29.145 * 0.697302765) + (-27.165 * 0.769965147) + \cdots + (\ldots_{(n)} * \ldots_{(n)}) = -1.45802338$$

Matriks *moore-penrose pseudo inverse* pada Tabel 4.13 merupakan hasil dari perkalian matriks *inverse* ordo hasil dari perkalian matriks *transpose* dari matriks fungsi aktifasi ordo 2x2 dengan matriks *transpose* fungsi aktifasi ordo 2x38 (Lihat Lampiran 9).

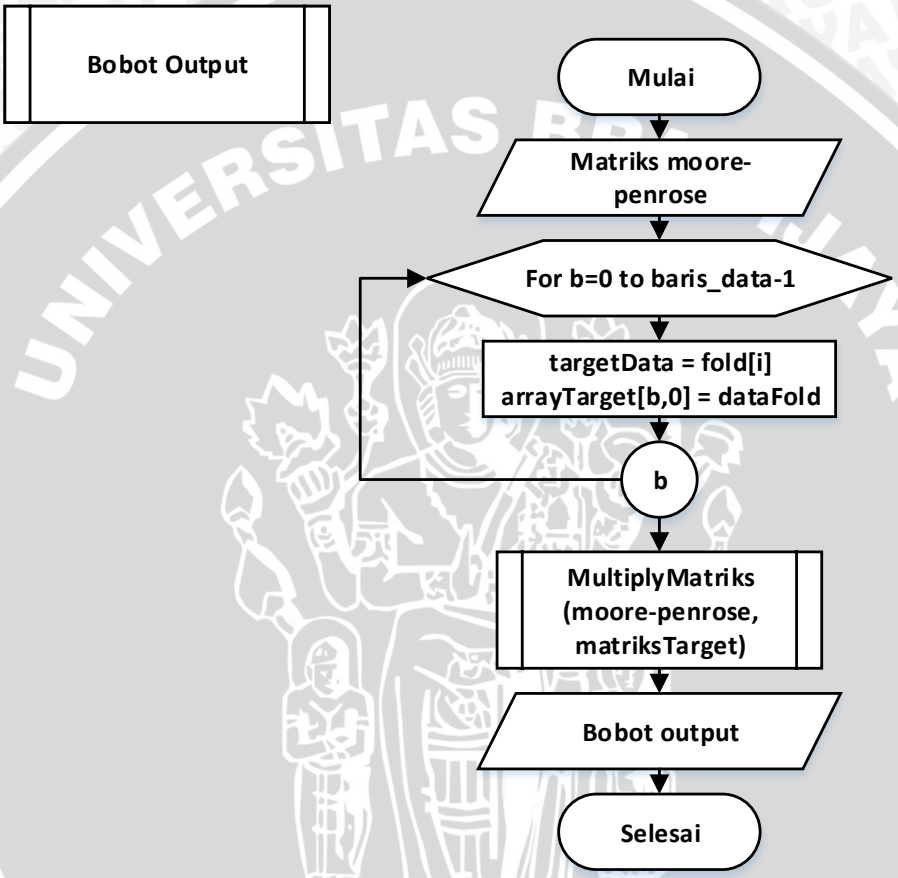
Tabel 4.13 Matriks *Moore-Penrose Pseudo Inverse*

H^+	1	2	...	38
1	-0.759598	1.200199	...	0.182587
2	0.7392429	-1.08748	...	-0.13955

Langkah 6: Hitung bobot *output* menggunakan Persamaan 6. Bobot *output* digunakan dalam proses *testing* untuk menentukan hasil keluaran. Dalam menghitung bobot *output* mengalikan matriks *moore-penrose pseudo inverse* dengan matriks target.

Matriks target dari data *training* yang ditransformasikan kedalam matriks ordo 38x1 diambil dari kolom kelas pada data *training*. Matriks target digunakan untuk mencari nilai bobot *output*. Gambar 4.11 merupakan diagram alur perhitungan bobot *output* dari hasil *training* menggunakan data *training fold*-1 dan *fold*-2. Berikut ini contoh perhitungan manualisasinya.

$$\beta_1 = (-1.45802338 * 1) + (-0.759597732 * 1) + \cdots + \left(\ldots_{(n)} * \ldots_{(n)}\right)$$
$$= 8.79524042$$



Gambar 4.11 *Flowchart* Bobot *Output*

Berdasarkan Gambar 4.11 diatas langkah – langkah proses mencari bobot *output* sebagai berikut.

a. Inputan proses ini yaitu matriks *moore-penrose* dengan matriks target kelas dari data *training*.

b. Proses perkalian matriks *moore-penrose* dengan matriks target menghasilkan bobot *output*.

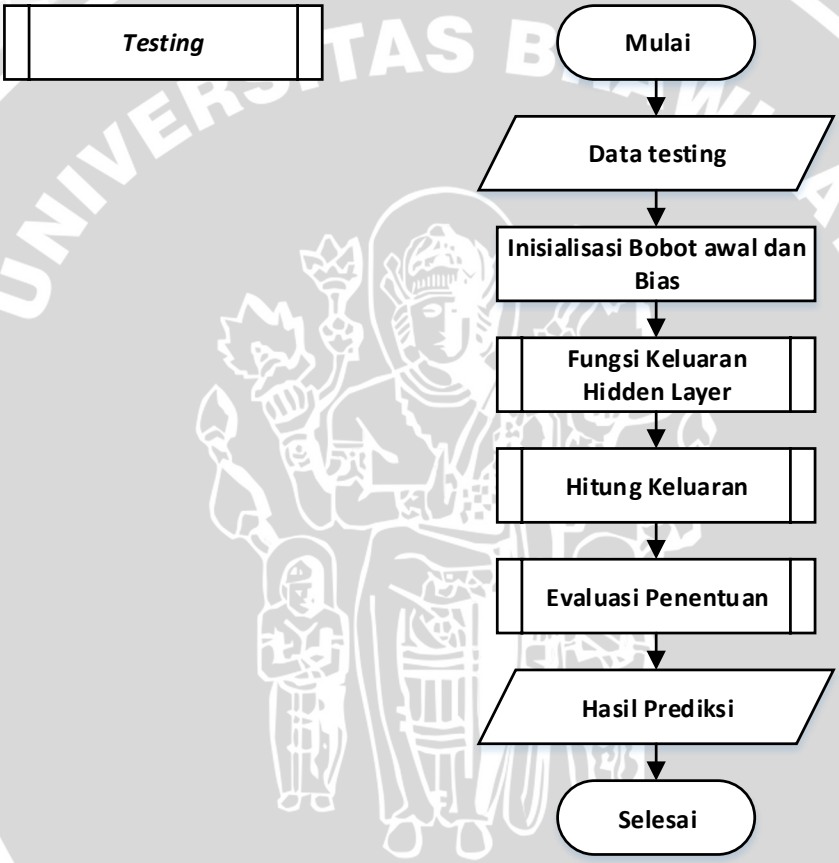
Tabel 4.14 Matriks Bobot *Output*

β	1
1	8.79524042

2	-5.638528651

Pada Tabel 4.14 merupakan matriks bobot *output* dari proses *training* menggunakan *fold*-1 sebagai *data training*. Matriks diatas hasil dari perkalian matriks *moore-penrose pseudo inverse* ordo 2x2 dengan matriks target ordo 38x1 menghasilkan matriks bobot*output* 2x1.

Langkah 7: Melakukan proses *testing* untuk menghitung keluaran. Proses *testing* menggunakan data *fold*-3. Perhitungan yang sama dilakukan pada proses *training* mulai langkah 1 sampai langkah 4. Berikut langkah – langkah proses *testing*. Langkah – langkah proses *testing* ditunjukkan pada Gambar 4.12.



Gambar 4.12 *Flowchart* Proses *Testing*

Berdasarkan Gambar 4.12 langkah – langkah proses *testing* sebagai berikut:

a. Inisialisasi bobot awal dan bias menggunakan Persamaan 1 dan 2. Bobot awal dan bias diambil secara acak dengan rentang nilai 0 sampai 1.
b. Menghitung keluaran dari setiap *hidden node* menggunakan Persamaan 3 yang hasilnya berupa matriks dan menggunakan fungsi aktivasi untuk memetakan hasil keluaran tersebut. Fungsi aktifasi yang digunakan pada proses manualisasi yaitu *Sigmoid Biner* pada Persamaan 4.
c. Hitung keluaran *output* menggunakan Persamaan 7.

d. Evaluasi hasil keluaran dengan menghitung jarak setiap Kelas. Dari jarak tersebut diperoleh nilai minimal. Nilai minimal itulah merupakan hasil dari penentuan Kelas pada sistem.

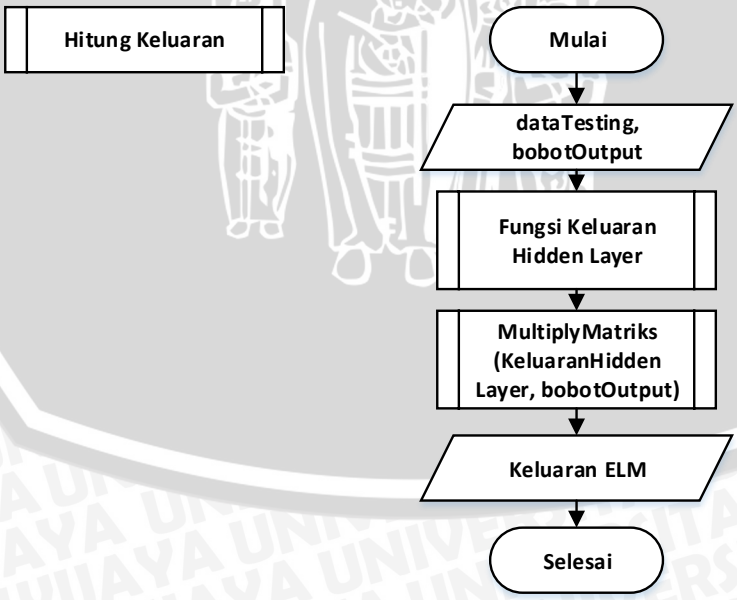
Langkah 7.1: hitung matriks keluaran dari setiap *hidden layer*. Nilai bobot awal dan bias yang digunakan sama dengan proses *training*. Sedangkan untuk *input layer* menggunakan data *testing fold*-3. Langkah – langkah perhitungan ditunjukkan pada Gambar 4.6. Hasil perhitungan matriks keluaran *hidden layer* pada Lampiran 10.

Matriks keluaran *hidden layer* hasil dari perkalian matriks *input* pada *fold*-3 ordo 22x7 dengan matriks bobot ordo 7x2 menghasilkan matriks keluaran *hidden layer* ordo 22x2 yang setiap kolomnya dijumlahkan dengan kolom yang sama pada bias.

Langkah 7.2: hitung matriks keluaran *hidden layer* menggunakan fungsi aktifasi *Sigmoid Biner* untuk memetakan nilainya. Langkah – langkah perhitungan mencari keluaran *hidden layer* ditunjukkan pada Gambar 4.1. Hasil perhitungan matriks fungsi aktifasi dapat dilihat pada Lampiran 11.

Hasil dari perhitungan fungsi aktifasi *Sigmoid Biner* memetakan nilai dari matriks keluaran *hidden layer fold*-3. Matriks diatas memiliki nilai yang lebih kecil sehingga memudahkan dalam operasi pada matriks selanjutnya.

Langkah 7.3: Menghitung keluaran. Perhitungan ini mencari nilai *ouput* menggunakan bobot *output* hasil proses *training*. Langkah – langkah perhitungan ditunjukkan pada Gambar 4.12. Berikut perhitungan mencari keluaran menggunakan Persamaan 3.



Gambar 4.13 *Flowchart* Hitung Keluaran ELM

Berdasarkan Gambar 4.13 diatas langkah – langkah proses menghitung keluaran ELM sebagai berikut.

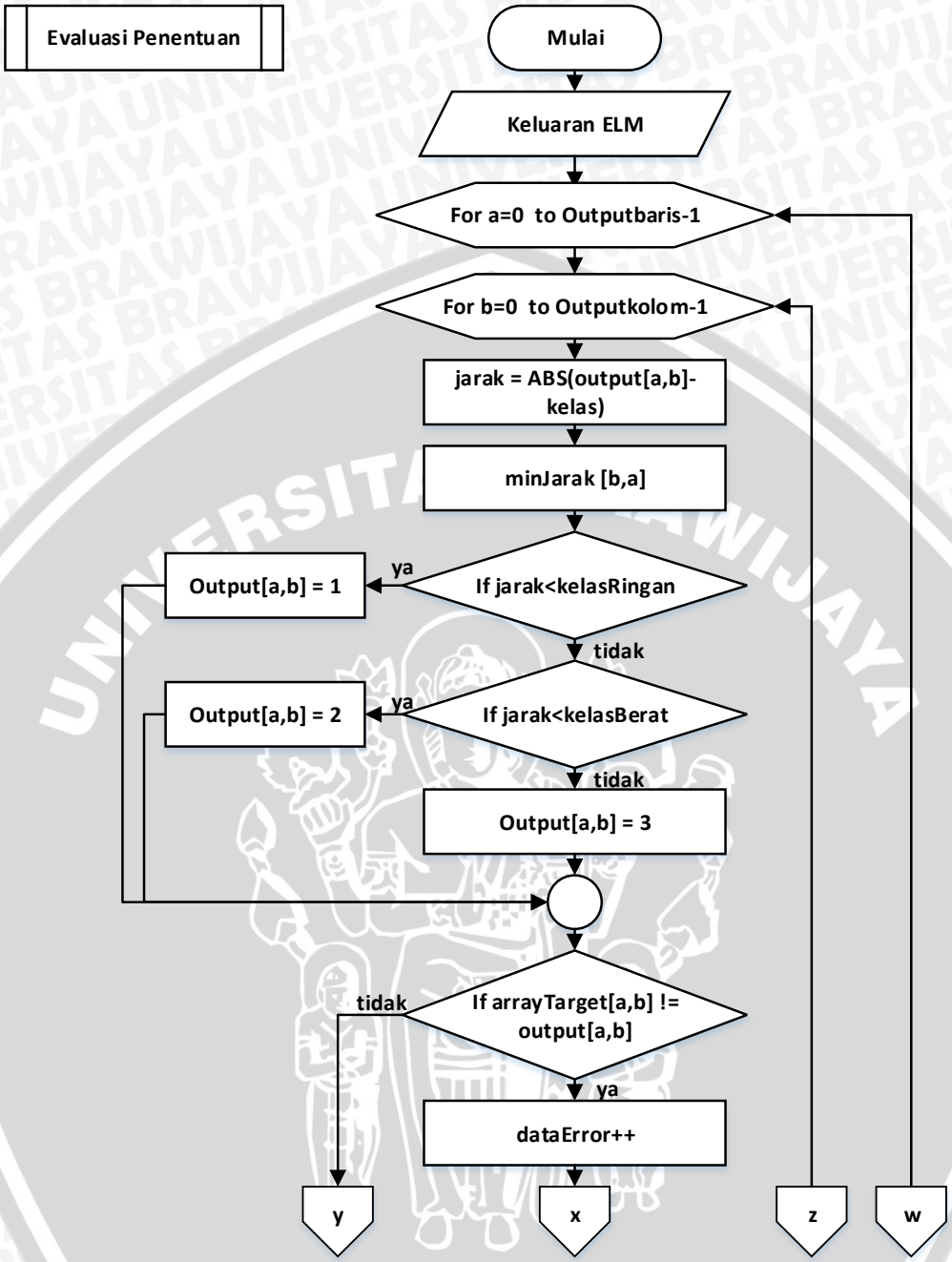
a. Inputan yang digunakan yaitu data *testing* dan bobot *output* hasil proses *training* sebelumnya.
b. Melakukan proses perhitungan mencari fungsi keluaran *hidden layer* menggunakan data *testing* dan bobot awal yang telah didefinisikan.
c. Melakukan proses perkalian fungsi keluaran *hidden layer* dengan bobot *output* hasil dari proses *training* sebelumnya, sehingga diperoleh hasil keluaran ELM.

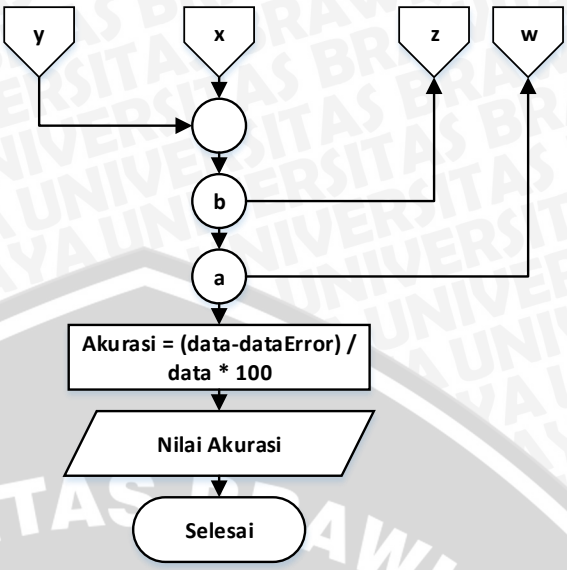
Berdasarkan Persamaan 7, contoh perhitungan manual keluaran ELM sebagai berikut.

$$\boldsymbol{O_1} = (0.730799 * 8.79524042) + (0.823931 * -5.638528651) + \cdots + (\ldots_{(n)} * \ldots_{(n)}) = 1.781788$$

Pada Tabel 4.10 hasil keluaran proses *testing* berupa matriks ordo 22x2 pada kolom 1. Nilai matriks diatas akan dihitung nilai selisih dari masing – masing Kelas pengelompokan, kemudian didapatkan nilai *minimum* dari ketiga kelas tersebut. Nilai *minimum* akan dikonversi kedalam Kelas dengan mencocokkan setiap nilai *minimum* dengan setiap nilai selisih. Alur proses perhitungan evaluasi ditunjukkan pada Gambar 4.14. Hasil perhitungan akhir pada Tabel 4.10 dari proses manualisasi dengan prediksi Kelas.

Perhitungan akurasi dilakukan dengan mencari data *error* antara target dengan hasil keluaran ELM. Dari nilai tersebut diperoleh data yang sama antara keduanya dan dibandingkan dengan keseluruhan data yang ada. Hasil perbandingan tersebut kemudian diambil nilai persentase yang merupakan tingkat keakurasian algoritma ELM dalam menentukan kualitas air sungai.


Evaluasi Penentuan
Mulai
Keluaran ELM
For a=0 to Outputbaris-1
For b=0 to Outputkolom-1
jarak = ABS(output[a,b]-kelas)
minJarak [b,a]
If jarak<kelasRingan
ya
Output[a,b] = 1
tidak
If jarak<kelasBerat
ya
Output[a,b] = 2
tidak
Output[a,b] = 3
If arrayTarget[a,b] != output[a,b]
tidak
ya
dataError++
y
x
z
w



Gambar 4.14 *Flowchart* Proses Evaluasi Penentuan

Berdasarkan Gambar 4.14 diatas langkah – langkah proses evaluasi penentuan keluaran ELM sebagai hasil dengan akurasi sebagai berikut.

a. Menghitung jarak antara keluaran ELM dengan setiap kelas sehingga diperoleh nilai absolut dari jarak tersebut.
b. Dari ketiga jarak kelas tersebut diambil nilai minimal, yaitu jarak terdekat antara keluaran dengan kelas.
c. Memberikan nilai sesuai kelasnya dari nilai jarak minimal tersebut, sehingga diperoleh hasil penentuan dari sistem.
d. Mencari data tidak sama antara hasil dari sistem dengan hasil menggunakan STORET.
e. Menghitung akurasi sistem dengan menggunakan Persamaan 8, sehingga diperoleh nilai akurasi sistem dalam memberikan solusi permasalahan.

Langkah 7.4: Menghitung nilai absolut jarak antara *output* dengan setiap kelas. Terdapat 3 kelas yaitu kelas 1 (tercemar ringan), kelas 2 (tercemar sedang), dan kelas 3 (tercemar berat). Berikut contoh perhitungan manual nilai absolut jarak antara *output* ke-1 dengan tiap kelas.

Jarak $O_1 \rightarrow kelas\ 1 : x = ABS(O_1 - 1) = ABS(1.781788 - 1) = 0.781788$

Jarak $O_1 \rightarrow kelas\ 2 : x = ABS(O_2 - 2) = ABS(1.781788 - 2) = 0.218211748$

Jarak $O_1 \rightarrow kelas\ 3 : x = ABS(O_3 - 3) = ABS(1.781788 - 3) = 1.218212$

Berdasarkan perhitungan diatas diambil nilai yang paling kecil yaitu jarak yang paling dekat antara *output* dengan kelas. Pada contoh perhitungan manualisasi *O* ke-1 diperoleh jarak terkecil dengan kelas 2. Hasilnya, *O* ke-1 termasuk dalam kelas 2 (tercemar sedang).

Tabel 4.15 Hasil Penentuan Kualitas Air Sungai

O	Kelas 1	Kelas 2	Kelas 3	nilai minimal	prediksi	Kelas
1.781788	0.781788	0.218212	1.218212	0.218211748	2	1
2.315292	1.315292	0.315292	0.684708	0.315292105	2	2
2.427922	1.427922	0.427922	0.572078	0.427922065	2	2
2.222625	1.222625	0.222625	0.777375	0.222625009	2	2
1.885551	0.885551	0.114449	1.114449	0.114449029	2	2
1.93273	0.93273	0.06727	1.06727	0.067270292	2	2
2.18356	1.18356	0.18356	0.81644	0.183559776	2	2
1.883385	0.883385	0.116615	1.116615	0.116615334	2	2
1.986536	0.986536	0.013464	1.013464	0.013464232	2	2
2.187932	1.187932	0.187932	0.812068	0.187932194	2	2
2.211232	1.211232	0.211232	0.788768	0.211232165	2	2
2.236793	1.236793	0.236793	0.763207	0.236792806	2	2
1.979603	0.979603	0.020397	1.020397	0.020396551	2	2
1.783771	0.783771	0.216229	1.216229	0.216229118	2	2
2.134168	1.134168	0.134168	0.865832	0.134168036	2	2
1.971316	0.971316	0.028684	1.028684	0.028684227	2	2
1.90168	0.90168	0.09832	1.09832	0.098320303	2	2
2.356018	1.356018	0.356018	0.643982	0.356017933	2	3
2.250074	1.250074	0.250074	0.749926	0.250073992	2	3
2.337347	1.337347	0.337347	0.662653	0.337346665	2	3
2.554603	1.554603	0.554603	0.445397	0.445397388	3	3
2.544867	1.544867	0.544867	0.455133	0.455132734	3	3

Berdasarkan Tabel 4.15 hasil prediksi dari sistem dibandingkan dengan hasil dari pakar yang menggunakan STORET. Akurasi dihitung menggunakan Persamaan 8, berikut contoh perhitungannya.

$$Akurasi = \frac{22-4}{22} x100\% = 81.818\%$$

Hasil yang ditunjukkan pada Tabel 4.15 merupakan contoh perhitungan menggunakan data *training fold*-1 dan *fold*-2 sebagai dan data *testing* pada *fold*-3. Tingkat akurasi yang dicapai adalah 81.818 %. Sedangkan, untuk percobaan menggunakan *fold*-2 sebagai data *testing* akurasi yang diperoleh adalah 78,947% dan *fold*-1 sebagai data *testing* adalah 73,684%. Rata – rata akurasi yang dicapai dengan nilai k = 3, jumlah *hidden layer* = 2, dan menggunakan fungsi aktivasi sigmoid biner adalah 78.15%.

4.3 Skenario Pengujian Algoritma ELM

Pengujian dilakukan untuk mengetahui hasil dari pemecahan permasalahan dari algoritma ELM. Adapun skenario pengujian yang dilakukan dalam penelitian ini sebagai berikut:

1. Pengujian nilai k
2. Pengujian jumlah *hidden node*

4.3.2 Pengujian Nilai *k*

Pengujian nilai k dilakukan untuk mengetahui pembagian *data training* dan *data testing* yang optimal dari metode *K-Fold Cross Validation* sehingga menghasilkan solusi terbaik. Dalam pengujian ini, semua data secara bergantian akan menjadi *data traning* dan hanya satu lainnya menjadi *data testing* untuk memperoleh hasil akurasi optimal. Banyaknya percobaan ditentukan oleh nilai k dengan minimal 2 dan maksimal 5. Pada Tabel 4.24 merupakan contoh skenario pengujian banyaknya nilai k.

Tabel 4.16Skenario Pengujian Nilai k

Nilai *k*	Data *Testing* (%)					Rata – Rata (%)
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
2						
3						
4						
5						

Pada Tabel 4.24 akan diperoleh hasil akurasi dari setiap percobaan banyaknya nilai k. Setiap nilai k akan menentukan banyaknya *fold* dan setiap *fold* secara bergantian akan digunakan sebagai *data training* dan satu yg lain sebagai *data testing*. Ketentuan penggunaan *fold* secara bergantian yaitu data pada proses *testing* hanya satu dari keseluruhan *fold*.

4.3.3 Pengujian Jumlah *Hidden Node*

Pengujian *hidden node* digunakan untuk mengetahui jumlah *hidden node* terbaik sehingga dihasilkan akurasi optimal. *Hidden node* digunakan untuk menentukan bobot awal dan bias untuk proses perhitungan ELM. Hal ini, memungkinkan perbedaan banyaknya *hidden node* mempengaruhi hasil akurasi pada ELM. Banyaknya jumlah *hidden node* rentang nilai 2 sampai 7. Pengujian jumlah *hidden node* ini digunakan pada nilaik k yang terbaik hasil dari pengujian sebelumnya. Pada Tabel 4.25 akan ditunjukkan tabel pengujian jumlah *hidden node* dan hasil akurasi terbaik.

Tabel 4.17 Skenario Pengujian Jumlah *Hidden Node*

Jumlah *Hidden Node*	Nilai *k* = 5					Rata – Rata (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
2						

3						
4						
5						
6						
7						

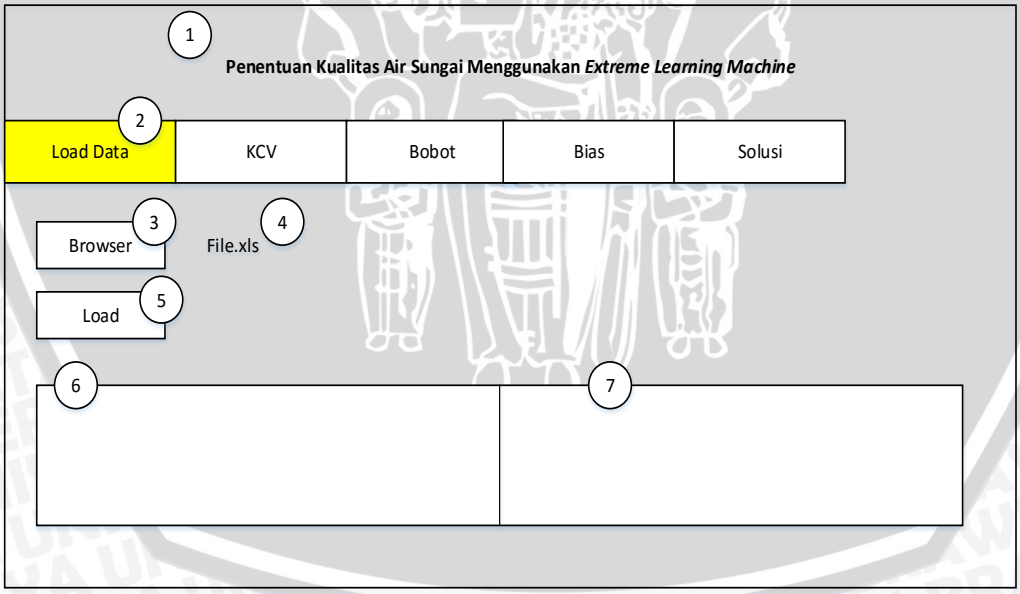
Hasil akurasi dari setiap banyaknya *hidden node* pada Tabel 4.25 akan diambil akurasi yang paling tinggi. Jumlah yang memiliki akurasi yang paling tinggi itulah yang akan digunakan dalam inisialisasi bobot awal dan bias.

4.4 Perancangan Antarmuka

Perancangan antar muka dibuat untuk realisasidari sistem yang sebenarnya dari implementasi pada penelitian ini. Implementasi antarmuka yang dibangun terdiri dari 6 antarmuka, yaitu *Load* Data, Normalisasi, Bobot, Bias, dan Solusi.

4.4.1 Perancangan Antarmuka *Load* Data

Antarmuka *load* data merupakan antarmuka yang digunakan pengguna untuk melakukan *entry* data dari *storage* berupa *file* dengan ekstensi .xls. Berikut ini merupakan rancangan antarmuka antarmuka *load*data ditunjukkan pada Gambar 4.15 dibawah ini.



Gambar 4.15 Perancangan Antarmuka *Load* Data

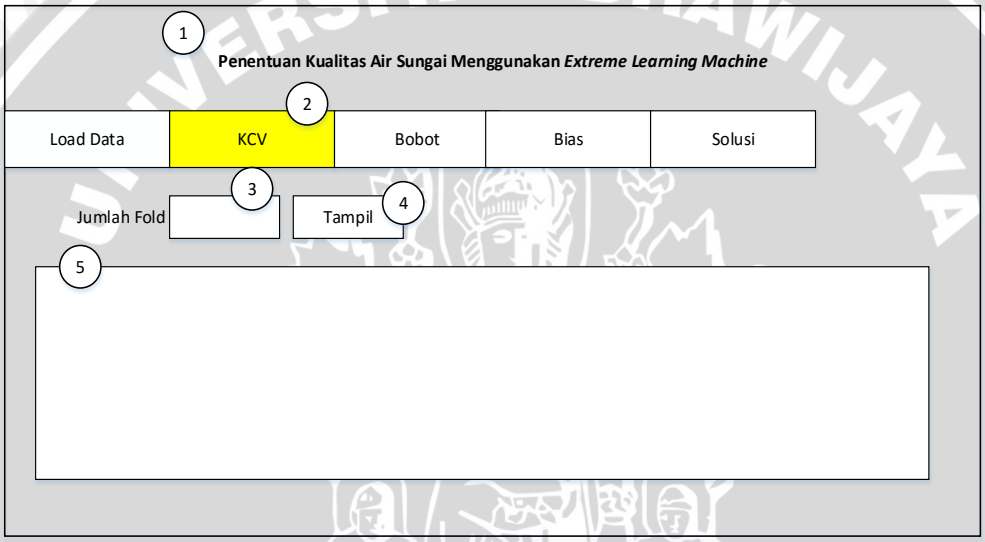
Penjelasan tentang rancangan antarmuka antarmuka*load* data pada Gambar 4.15 sebagai berikut:

1. *Header* program

2. Menu bar pada program berwarna kuning menandakan menu Load Data sedang aktif.
3. *Button* Browser untuk memilih *file* yang akan di *load* pada *storage*.
4. *TextView* menunjukkan nama *file* yang dipilih.
5. *Button*Load untuk melakukan proses *input* data pada program.
6. *TextBox* menampilkan data yang telah berhasil di *load*.
7. *TextBox* menampilkan data yang telah berhasil dinormalisasi.

4.4.2 Perancangan AntarmukaK-*Fold Cross Validation*

Antarmukak-*fold cross validation* merupakan antarmuka yang menampilkan hasil pembagian data yang telah dinormalisasi dari data masukan pengguna sebelumnya. Berikut ini merupakan rancangan antarmuka antarmukak-*fold cross validation* ditunjukkan pada Gambar 4.16 dibawah ini.



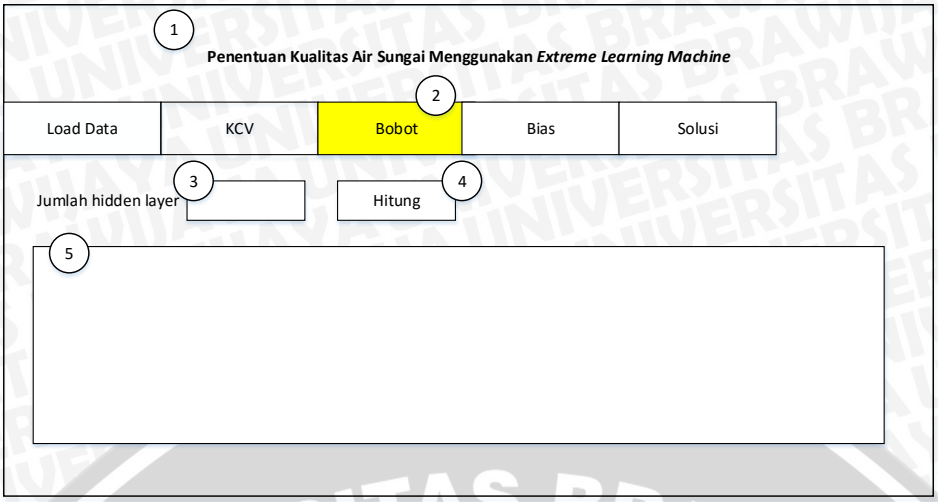
Gambar 4.16 Perancangan Antarmuka K-*Fold Cross Validation*

Penjelasan tentang rancangan antarmuka antarmukanormalisasi pada Gambar 4.16 sebagai berikut:

1. *Header* program
2. Menu bar pada program berwarna kuning menandakan menu KCV sedang aktif.
3. *TexBox*untuk memasukkan nilai *fold* yang diinginkan.
4. *Button*Tampil untuk melakukan proses pembagian data.
5. *TexBox* menampilkan hasil pembagian data.

4.4.3 Perancangan Antarmuka Bobot

Antarmuka bobot merupakan antarmuka yang menampilkan bobot secara *random*. Berikut ini merupakan rancangan antarmuka antarmuka bobot ditunjukkan pada Gambar 4.17 dibawah ini.



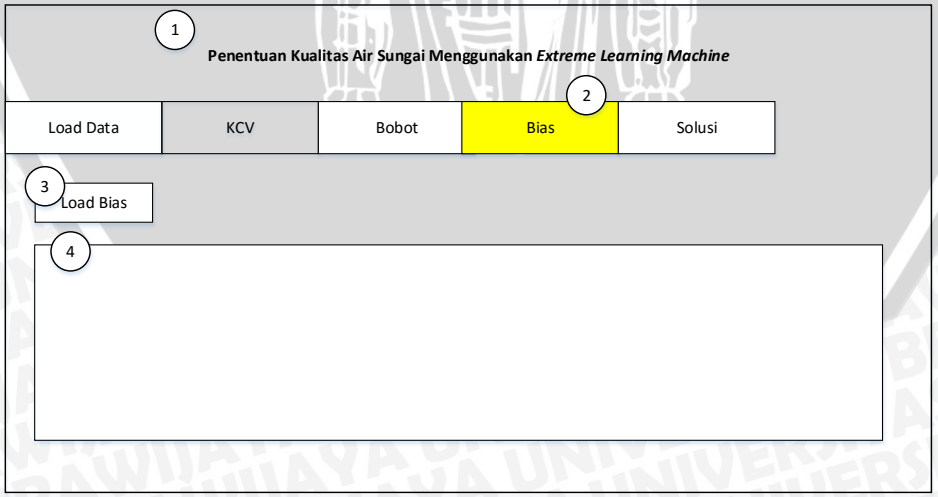
Gambar 4.17 Perancangan Antarmuka Bobot

Penjelasan tentang rancangan antarmuka antarmukabobot pada Gambar 4.17 sebagai berikut:

1. *Header* program
2. Menu bar pada program berwarna kuning menandakan menu Bobot sedang aktif.
3. *TextBox*untuk memasukkan jumlah hidden layer untuk membentuk bobot awal dan bias.
4. *Button*Hitung untuk melakukan proses menghitung bobot awal.
5. *TextBox* menampilkan bobot awal.

4.4.4 Perancangan Antarmuka Bias

Antarmuka bias merupakan antarmuka yang menampilkan nilai bias secara *random*. Berikut ini merupakan rancangan antarmuka antarmuka bias ditunjukkan pada Gambar 4.18 dibawah ini.



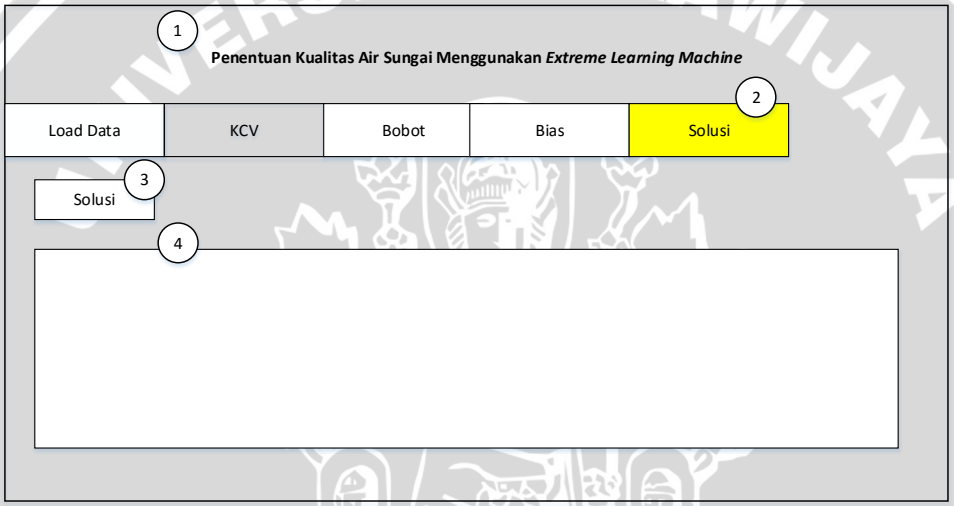
Gambar 4.18 Perancangan Antarmuka Bias

Penjelasan tentang rancangan antarmuka antarmukabias pada Gambar 4.18 sebagai berikut:

1. *Header* program
2. Menu bar pada program berwarna kuning menandakan menu Bias sedang aktif.
3. *Button* Bias Awal untuk memasukkan *loadfile* bias.
4. *TextBox* menampilkan hasil bias.

4.4.5 Perancangan Antarmuka Solusi

Antarmuka solusi merupakan antarmuka yang menampilkan hasil penentuan dari proses perhitungan algoritma ELM. Berikut ini merupakan rancangan antarmuka antarmuka bias ditunjukkan pada Gambar 4.19 dibawah ini.



Gambar 4.19 Perancangan Antarmuka Solusi

Penjelasan tentang rancangan antarmuka antarmukasolusi pada Gambar 4.19 sebagai berikut:

1. *Header* program
2. Menu bar pada program berwarna kuning menandakan menu Solusi sedang aktif.
3. *Button*Solusi untuk melakukan perhitungan hasil keluaran.
4. *TextBox* menampilkan hasil solusi.

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem berdasarkan analisiskebutuhan dan proses perancangan sistem yang telah dibuat. Pembahasan pada bab ini terdiri dari penjelasan algoritma *extreme learning machine*serta implementasi antarmukasistem.

5.1 Implementasi Algoritma *Extreme Learning Machine*

Implementasi algoritma merupakan hasil dari perancangan sistem penentuan kualitas air sungai menggunakan algoritma *extreme learning machine* ke dalam kode program. Penentuan kualitas air sungai ini terdiri dari 3 proses, yaitu proses normalisasi data, proses *k-fold cross validation*, dan proses *extreme learning machine*. Proses ELM sendiri terdiri dari proses menghitung fungsi keluaran *hidden layer*, proses menghitung matriks *moore-penrose pseudo invers*, proses menghitung bobot *output* kemudian masuk proses *testing*. Proses *testing* terdiri dari proses menghitung *output*, proses evaluasi penentuan. Proses evaluasi penentuan digunakan untuk menentukan hasil diagnosa dari *output* ELM. Proses evaluasi penentuan meliputi proses menghitung jarak antara *output* dengan setiap kelas, proses mencari nilai minimum dari jarak *output* dengan setiap klas, proses menghitung akurasi dan rata – rata akurasi yang diperoleh.

5.1.1 Implementasi Normalisasi Data

Implementasi normalisasi data merupakan proses awal sebelum proses utama penentuankualitas air sungai. Proses ini bertujuan untuk menyamakan rentang nilai setiap atribut agar berada pada interval 0 sampai dengan 1. Implementasi normalisasi data ditunjukkan pada Kode Program 5.1 sebagai berikut:

Kode Program 5.1 Implementasi Normalisasi Data

```
1.  // method hitung normalisasi
2.          private void CountNormalisasi()
3.          {
4.              if (this.CheckDatas()) // kondisi cek data
5.              {
6.                  DataHelper.Current.IsNormalisasiCounted = true;
7.                  //nilai max dan min Tss
8.                  var minTss = this.WaterDatas.Min(x => x.Tss);
9.                  var maxTss = this.WaterDatas.Max(x => x.Tss);
10.                 var medTss = maxTss - minTss;
11.
12.                //nilai max dan min Bod
13.                 var minBod = this.WaterDatas.Min(x => x.Bod);
14.                 var maxBod = this.WaterDatas.Max(x => x.Bod);
15.                 var medBod = maxBod - minBod;
16.
17.                 //nilai max daan min Cod
18.                 var minCod = this.WaterDatas.Min(x => x.Cod);
19.                 var maxCod = this.WaterDatas.Max(x => x.Cod);
20.                 var medCod = maxCod - minCod;
```

```
21.
22.                 //nilai max daan min Do
22.                 var minDo = this.WaterDatas.Min(x => x.Do);
23.                 var maxDo = this.WaterDatas.Max(x => x.Do);
24.                 var medDo = maxDo - minDo;
25.
26.                 //nilai max daan min pH
27.                 var minpH = this.WaterDatas.Min(x => x.Ph);
28.                 var maxpH = this.WaterDatas.Max(x => x.Ph);
29.                 var medpH = maxpH - minpH;
30.
31.                 //nilai max daan min Fenol
32.                 var minFenol = this.WaterDatas.Min(x => x.FenolDec);
33.                 var maxFenol = this.WaterDatas.Max(x => x.FenolDec);
34.                 var medFenol = maxFenol - minFenol;
35.
36.                 //nilai max daan min Oil and Fat
37.                 var minOilAndFat = this.WaterDatas.Min(x =>
38. x.OilAndFat);
39.                 var maxOilAndFat = this.WaterDatas.Max(x =>
40. x.OilAndFat);
41.                 var medOilAndFat = maxOilAndFat - minOilAndFat;
42.
43.                 // hitung normalisasi tiap fitur
44.                 foreach (var waterData in this.WaterDatas)
45.                 {
46.                     waterData.NormalisasiTss = (waterData.Tss -
47. minTss) / medTss;
48.                     waterData.NormalisasiBod = (waterData.Bod -
49. minBod) / medBod;
50.                     waterData.NormalisasiCod = (waterData.Cod -
51. minCod) / medCod;
52.                     waterData.NormalisasiDo = (waterData.Do - minDo) /
53. medDo;
54.                     waterData.NormalisasiPh = (waterData.Ph - minpH) /
55. medpH;
56.                     waterData.NormalisasiFenol = (waterData.FenolDec -
57. minFenol) / medFenol;
58.                     waterData.NormalisasiOilAndFat =
59. (waterData.OilAndFat - minOilAndFat) / medOilAndFat;
60.                 }
61.                 this.OnPropertyChanged(nameof(this.WaterDatas));
62.             }
63.         }
```

Penjelasan Kode Program 5.1 tentang implementasi normalisasi data adalah sebagai berikut:

Baris 4 s.d 6	:	Cek kondisi data tersedia.
Baris 7 s.d 42	:	Menghitung nilai maksimum dan nilai minimum dari setiap atribut data
Baris 43 s.d 60	:	Menghitung nilai normalisasi setiap atribut data menggunakan Persamaan 10.
Baris 61 s.d 62	:	Update nilai normalisasi pada antarmuka

5.1.2 Implementasi K-*Fold Cross Validation*

Implementasi *k-fold cross validation* merupakan proses membagi data set ke dalam data *training* dan data *testing*. Proses ini bertujuan untuk membagi data *training* dan data *testing* yang ideal dari setiap klas yang ada supaya diperoleh pembagian yang terbaik dengan akurasi yang baik nantinya. Implementasi *k-fold cross validation* ditunjukkan pada Kode Program 5.2 sebagai berikut:

Kode Program 5.2 Implementasi K-*Fold Cross Validation*

```
1.  // method menghitung pembagian Fold
2.          private void CountKcv()
3.          {
4.              if (CheckDatas())
5.              {
6.                  // kondisi cek data
7.                  _WaterDatas = new
8.  List<WaterData>(DataHelper.Current.Datas);
9.
10.                 // inisialisasi variabel untuk menyimpan Fold
11.                 Fold1 = new AsyncObservableCollection<WaterData>();
12.                 Fold2 = new AsyncObservableCollection<WaterData>();
13.                 Fold3 = new AsyncObservableCollection<WaterData>();
14.                 Fold4 = new AsyncObservableCollection<WaterData>();
15.                 Fold5 = new AsyncObservableCollection<WaterData>();
16.
17.                 // data akan dibagi menjadi sebanyak variabel k
18.                 var k = DataHelper.Current.KValue;
19.
20.                 // menghitung jumlah keselurahan data dari setiap
21.  klas (klas 1, klas 2, dan klas 3)
22.                 var countRingan =
22.                     _WaterDatas.Count(x => x.Klas.Equals("Ringan",
23.  StringComparison.OrdinalIgnoreCase));
24.                 var countSedang =
25.                     _WaterDatas.Count(x => x.Klas.Equals("Sedang",
26.  StringComparison.OrdinalIgnoreCase));
27.                 var countBerat = _WaterDatas.Count(x =>
28.  x.Klas.Equals("Berat", StringComparison.OrdinalIgnoreCase));
29.
30.                 // membagi data dari tiap klas
31.                 var countRinganEachKlas = (int)
32.  Math.Floor(countRingan/(double) k);
33.                 var countSedangEachKlas = (int)
34.  Math.Floor(countSedang/(double) k);
35.                 var countberatEachKlas = (int)
36.  Math.Floor(countBerat/(double) k);
37.
38.                 // menyimpan data untuk setiap data yang sudah
39.  diperoleh
40.                 DataHelper.Current.RinganDataCount =
41.  countRinganEachKlas;
42.                 DataHelper.Current.SedangDataCount =
43.  countSedangEachKlas;
44.                 DataHelper.Current.BeratDataCount =
45.  countberatEachKlas;
46.
47.                 // random data pengambilan setiap fold
```

```
48.                 var random = new Random();
49.                 if (k == 2)
50.                 {
51.                     // menaruh data yang diambil dari setiap klas ke
52. dalam Fold 1
53.                     Fold1.AddRange(
54.                         _WaterDatas.Where(x =>
55. x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
56.                             .OrderBy(x =>
57. random.Next()).Take(countRinganEachKlas));
58.                     Fold1.AddRange(
59.                         _WaterDatas.Where(x =>
60. x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
61.                             .OrderBy(x =>
62. random.Next()).Take(countSedangEachKlas));
63.                     Fold1.AddRange(
64.                         _WaterDatas.Where(x => x.Klas.Equals("Berat",
65. StringComparison.OrdinalIgnoreCase))
66.                             .OrderBy(x =>
67. random.Next()).Take(countberatEachKlas));
68.
69.                     // data yang sudah di taruh kedalam Fold 1 akan
70. di remove dari data set
71.                     _WaterDatas.RemoveAll(x => Fold1.Contains(x));
72.
73.                     Fold2.AddRange(_WaterDatas.OrderBy(x =>
74. x.KlasWeight));
75.
76.                     _WaterDatas.RemoveAll(x => Fold2.Contains(x));
77.
78.                     Fold1Visibility = true;
79.                     Fold2Visibility = true;
80.                     Fold3Visibility = false;
81.                     Fold4Visibility = false;
82.                     Fold5Visibility = false;
83.                 }
84.                 else if (k == 3)
85.                 {
86.                     // menaruh data yang diambil dari setiap klas ke
87. dalam Fold 1
88.                     Fold1.AddRange(
89.                         _WaterDatas.Where(x =>
90. x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
91.                             .OrderBy(x =>
92. random.Next()).Take(countRinganEachKlas));
93.                     Fold1.AddRange(
94.                         _WaterDatas.Where(x =>
95. x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
96.                             .OrderBy(x =>
97. random.Next()).Take(countSedangEachKlas));
98.                     Fold1.AddRange(
99.                         _WaterDatas.Where(x => x.Klas.Equals("Berat",
100. StringComparison.OrdinalIgnoreCase))
101.                             .OrderBy(x =>
102. random.Next()).Take(countberatEachKlas));
103.
104.                     // data yang sudah di taruh kedalam Fold 1 akan
105. di remove dari data set
106.                     _WaterDatas.RemoveAll(x => Fold1.Contains(x));
107.
```

```
                    // menaruh data kedalam Fold 2
                    Fold2.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countRinganEachKlas));
                    Fold2.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countSedangEachKlas));
                    Fold2.AddRange(
                        _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                    // data yang sudah di taruh kedalam Fold 1 dan
Fold 2 akan di remove dari data set
                    _WaterDatas.RemoveAll(x => Fold2.Contains(x));

                    Fold3.AddRange(_WaterDatas.OrderBy(x =>
x.KlasWeight));

                    _WaterDatas.RemoveAll(x => Fold3.Contains(x));

                    Fold1Visibility = true;
                    Fold2Visibility = true;
                    Fold3Visibility = true;
                    Fold4Visibility = false;
                    Fold5Visibility = false;
                }
                else if (k == 4)
                {
                    // menaruh data yang diambil dari setiap klas ke
dalam Fold 1
                    Fold1.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countRinganEachKlas));
                    Fold1.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countSedangEachKlas));
                    Fold1.AddRange(
                        _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                    // data yang sudah di taruh kedalam Fold 1 akan
di remove dari data set
                    _WaterDatas.RemoveAll(x => Fold1.Contains(x));

                    // menaruh data kedalam Fold 2
                    Fold2.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan",
```

```
168.  StringComparison.OrdinalIgnoreCase)).OrderBy(x =>
169.  random.Next()).Take(countRinganEachKlas));
170.                      Fold2.AddRange(
171.                          _WaterDatas.Where(x =>
172.  x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
173.                              .OrderBy(x =>
174.  random.Next()).Take(countSedangEachKlas));
175.                      Fold2.AddRange(
176.                          _WaterDatas.Where(x => x.Klas.Equals("Berat",
177.  StringComparison.OrdinalIgnoreCase))
178.                              .OrderBy(x =>
179.  random.Next()).Take(countberatEachKlas));
180.
181.                      // data yang sudah di taruh kedalam Fold 1 dan
182.  Fold 2 akan di remove dari data set
183.                      _WaterDatas.RemoveAll(x => Fold2.Contains(x));
184.
185.                      Fold3.AddRange(
186.                          _WaterDatas.Where(x =>
187.  x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
188.                              .OrderBy(x =>
189.  random.Next()).Take(countRinganEachKlas));
190.                      Fold3.AddRange(
191.                          _WaterDatas.Where(x =>
192.  x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
193.                              .OrderBy(x =>
194.  random.Next()).Take(countSedangEachKlas));
195.                      Fold3.AddRange(
196.                          _WaterDatas.Where(x => x.Klas.Equals("Berat",
197.  StringComparison.OrdinalIgnoreCase))
198.                              .OrderBy(x =>
199.  random.Next()).Take(countberatEachKlas));
200.
201.                      _WaterDatas.RemoveAll(x => Fold3.Contains(x));
202.
203.                      Fold4.AddRange(_WaterDatas.OrderBy(x =>
204.  x.KlasWeight));
205.
206.                      _WaterDatas.RemoveAll(x => Fold4.Contains(x));
207.
208.                      Fold1Visibility = true;
209.                      Fold2Visibility = true;
210.                      Fold3Visibility = true;
211.                      Fold4Visibility = true;
212.                      Fold5Visibility = false;
213.                  }
214.                  else if (k == 5)
215.                  {
216.                      // menaruh data yang diambil dari setiap klas ke
217.  dalam Fold 1
218.                      Fold1.AddRange(
219.                          _WaterDatas.Where(x =>
220.  x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
221.                              .OrderBy(x =>
222.  random.Next()).Take(countRinganEachKlas));
223.                      Fold1.AddRange(
224.                          _WaterDatas.Where(x =>
225.  x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
226.                              .OrderBy(x =>
227.  random.Next()).Take(countSedangEachKlas));
```

```
                    Fold1.AddRange(
                        _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                    // data yang sudah di taruh kedalam Fold 1 akan
di remove dari data set
                    _WaterDatas.RemoveAll(x => Fold1.Contains(x));

                    // menaruh data kedalam Fold 2
                    Fold2.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countRinganEachKlas));
                    Fold2.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countSedangEachKlas));
                    Fold2.AddRange(
                        _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                    // data yang sudah di taruh kedalam Fold 1 dan
Fold 2 akan di remove dari data set
                    _WaterDatas.RemoveAll(x => Fold2.Contains(x));

                    Fold3.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countRinganEachKlas));
                    Fold3.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countSedangEachKlas));
                    Fold3.AddRange(
                        _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                    _WaterDatas.RemoveAll(x => Fold3.Contains(x));

                    Fold4.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Ringan", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countRinganEachKlas));
                    Fold4.AddRange(
                        _WaterDatas.Where(x =>
x.Klas.Equals("Sedang", StringComparison.OrdinalIgnoreCase))
                            .OrderBy(x =>
random.Next()).Take(countSedangEachKlas));
                    Fold4.AddRange(
```

```
                    _WaterDatas.Where(x => x.Klas.Equals("Berat",
StringComparison.OrdinalIgnoreCase))
                        .OrderBy(x =>
random.Next()).Take(countberatEachKlas));

                _WaterDatas.RemoveAll(x => Fold4.Contains(x));

                Fold5.AddRange(_WaterDatas.OrderBy(x =>
x.KlasWeight));

                _WaterDatas.RemoveAll(x => Fold5.Contains(x));

                Fold1Visibility = true;
                Fold2Visibility = true;
                Fold3Visibility = true;
                Fold4Visibility = true;
                Fold5Visibility = true;
            }
            // mengambil nilai Fold 1 dimasukkan kedalam array
            DataHelper.Current.Fold1Datas = Fold1.ToList();

            // inisialisasi variabel
            var row = Fold1.Count;
            var column = 7;
            var inputArrays = new double[row, column];

            // perulangan memasukkan nilai Fold 1 ke dalam array
            for (var i = 0; i < row; i++)
            {
                for (var j = 0; j < column; j++)
                {
                    inputArrays[i, j] =

Convert.ToDouble(Fold1[i].GetType().GetProperties()[j].GetValue(Fold1
[i], null));
                }
            }

            // membangun matriks Fold 1
            var matrixBuilderFold1 = Matrix<double>.Build;
            var matrixFold1 =
matrixBuilderFold1.DenseOfArray(inputArrays);
            DataHelper.Current.Fold1NormalisasiWaterDatas =
matrixFold1;

            // Fold 2
            DataHelper.Current.Fold2Datas = Fold2.ToList();
            inputArrays = new double[row, column];
            for (var i = 0; i < row; i++)
            {
                for (var j = 0; j < column; j++)
                {
                    inputArrays[i, j] =

Convert.ToDouble(Fold2[i].GetType().GetProperties()[j].GetValue(Fold2
[i], null));
                }
            }

            var matrixBuilderFold2 = Matrix<double>.Build;
```

```
                var matrixFold2 =
matrixBuilderFold2.DenseOfArray(inputArrays);
                DataHelper.Current.Fold2NormalisasiWaterDatas =
matrixFold2;

                if (k > 2)
                {
                    // Fold 3
                    DataHelper.Current.Fold3Datas = Fold3.ToList();
                    inputArrays = new double[Fold3.Count, column];
                    for (var i = 0; i < Fold3.Count; i++)
                    {
                        for (var j = 0; j < column; j++)
                        {
                            inputArrays[i, j] =
                                Convert.ToDouble(

Fold3[i].GetType().GetProperties()[j].GetValue(Fold3[i], null));
                        }
                    }
                    var matrixBuilderFold3 = Matrix<double>.Build;
                    DataHelper.Current.RinganDataCount =
Fold3.Count(x => x.KlasWeight == 1);
                    DataHelper.Current.SedangDataCount =
Fold3.Count(data => data.KlasWeight == 2);
                    DataHelper.Current.BeratDataCount =
Fold3.Count(data => data.KlasWeight == 3);
                    var matrixFold3 =
matrixBuilderFold3.DenseOfArray(inputArrays);
                    DataHelper.Current.Fold3NormalisasiWaterDatas =
matrixFold3;
                    if (k > 3)
                    {
                        // Fold 4
                        DataHelper.Current.Fold4Datas =
Fold4.ToList();
                        inputArrays = new double[Fold4.Count,
column];
                        for (var i = 0; i < Fold4.Count; i++)
                        {
                            for (var j = 0; j < column; j++)
                            {
                                inputArrays[i, j] =
                                    Convert.ToDouble(

Fold4[i].GetType().GetProperties()[j].GetValue(Fold4[i], null));
                            }
                        }
                        var matrixBuilderFold4 =
Matrix<double>.Build;
                        DataHelper.Current.RinganDataCount =
Fold4.Count(x => x.KlasWeight == 1);
                        DataHelper.Current.SedangDataCount =
Fold4.Count(data => data.KlasWeight == 2);
                        DataHelper.Current.BeratDataCount =
Fold4.Count(data => data.KlasWeight == 3);
                        var matrixFold4 =
matrixBuilderFold4.DenseOfArray(inputArrays);
                        DataHelper.Current.Fold4NormalisasiWaterDatas
= matrixFold4;
```

```
409.                         if (k > 4)
410.                         {
411.                             // Fold 5
412.                             DataHelper.Current.Fold5Datas =
413. Fold5.ToList();
414.                             inputArrays = new double[Fold5.Count,
415. column];
416.                             for (var i = 0; i < Fold5.Count; i++)
417.                             {
418.                                 for (var j = 0; j < column; j++)
419.                                 {
420.                                     inputArrays[i, j] =
421.                                         Convert.ToDouble(
422.
423. Fold5[i].GetType().GetProperties()[j].GetValue(Fold5[i], null));
424.                                 }
425.                             }
426.                             var matrixBuilderFold5 =
427. Matrix<double>.Build;
428.                             DataHelper.Current.RinganDataCount =
429. Fold5.Count(x => x.KlasWeight == 1);
430.                             DataHelper.Current.SedangDataCount =
431. Fold5.Count(data => data.KlasWeight == 2);
432.                             DataHelper.Current.BeratDataCount =
433. Fold5.Count(data => data.KlasWeight == 3);
434.                             var matrixFold5 =
435. matrixBuilderFold5.DenseOfArray(inputArrays);
436.
437. DataHelper.Current.Fold5NormalisasiWaterDatas = matrixFold5;
438.                         }
439.                     }
440.                 }
441.                 FoldVisibility = true;
442.             }
443.         }
444.
```

Penjelasan Kode Program 5.2 tentang implementasi *k-fold cross validation* adalah sebagai berikut:

Baris 1 s.d 3	:	Method pembagian data *fold*.
Baris 4 s.d 9	:	Cek ketersediaan data.
Baris 10 s.d 16	:	Inisialisasi variabel untuk menyimpan data pada setiap *fold*.
Baris 17 s.d 19	:	Inisialisasi variabel nilai *k* yang diterima dari inputan pengguna.
Baris 20 s.d 29	:	Proses menghitung keseluruhan data dari masing – masing kelas.
Baris 30 s.d 37	:	Mambagi data pada masing – masing kelas sebanyak nilai *k* untuk diletakkan pada setiap *fold*.
Baris 38 s.d 46	:	Menyimpan data yang telah dibagi dari setiap kelas kedalam variabel.
Baris 47 s.d 48	:	Inisialisasi variabel random untuk pengambilan data pada setiap *fold* secara acak.

Baris 49 s.d 305 : Proses pengambilan data *fold* secara acak dimulai dari nilai *k* sama dengan 2 sampai 5.

Baris 306 s.d 444 : Proses *input* kedalam array data yang telah dilakukan pengambilan pada setiap *fold* sesuai dengan nilai *k* yaitu 2 sampai 5.

5.1.3 Implementasi *Extreme Learning Machine*

Implementasi *extreme learning machine* merupakan proses utama penentuan kualitas air sungai. Proses ini terbagi menjadi dua proses, yaitu proses *training* dan proses *testing*. Proses *training* dilakukan untuk menghasilkan bobot *output* baru yang menghubungkan *hidden layer* dengan *output layer*. Proses *testing* dilakukan untuk memperoleh keluaran. Pada proses*testing* terdapat proses evaluasi penentuan dimana proses ini untuk menentukan diagnosa dari keluaran menjadi kelas kualitas air sungai.

Proses *training* menggunakan data *training* untuk memperoleh *output* baru. Sedangkan, proses *testing* menggunakan data *testing* untuk memperoleh keluaran. Selanjutnya, hasil keluaran tersebut masuk ke proses evaluasi penentuan. Proses ini melakukan perhitungan jarak *output* dengan setiap kelas dan mencari nilai minimalnya dan menghitung akurasi.

a. Implementasi Bobot Awal

Implementasi bobot awal dilakukan untuk memperoleh nilai bobot awal secara acak dengan rentang nilai 0 sampai dengan 1. Ukuran bobot awal ditentukan dengan rumus pada Persamaan 1. Berikut implementasi pembentukan bobot awal pada Kode Program 5.3.

Kode Program 5.3 Implementasi Pembentukan Bobot Awal

```
1.  // method load bobot
2.          private void LoadBobot()
3.          {
4.              //masukkan nilai input ke hidden layer
5.              DataHelper.Current.HiddenLayer =
6.  int.Parse(HiddenLayerCount);
7.              //untuk judul kolom di view
8.              var headerNames = new List<object>
9.              {
10.                 "Bobot 1",
11.                 "Bobot 2",
12.                 "Bobot 3",
13.                 "Bobot 4",
14.                 "Bobot 5",
15.                 "Bobot 6",
16.                 "Bobot 7"
17.             };
18.             var datas = new double[DataHelper.Current.HiddenLayer, 7];
19.             var random = new Random();
20.             //random array untuk bobot dg range 0-1
21.             for (var i = 0; i < DataHelper.Current.HiddenLayer; i++)
22.             {
22.                 for (var j = 0; j < 7; j++)
23.                 {
```

```
24.                     datas[i, j] = random.NextDouble();
25.                 }
26.             }
27.
28.             BobotDatas = new ArrayDataView(datas,
29. headerNames.ToArray());
30.             DataHelper.Current.BobotDatas = BobotDatas; // data
31. disimpan dalam variabel
```

Penjelasan Kode Program 5.3 tentang implementasi pembentukan bobot awal adalah sebagai berikut:

Baris 4 s.d 6	:	Inputan jumlah *hidden node*.
Baris 7 s.d 17	:	Pemberian nama pada *header* kolom pada bobot sesuai banyaknya *input node*.
Baris 18	:	Inisialisasi variabel jumlah *hidden layer* untuk menentukan baris dari bobot awal.
Baris 19	:	Inisialisasi variabel random untuk nilai acak antara 0 sampai 1.
Baris 20 s.d 26	:	Menghitung nilai bobot awal secara random dengan rentang nilai 0 sampai dengan 1.
Baris 27 s.d 31	:	Menyimpan nilai bobot awal kedalam variabel yang diperoleh secara acak dengan rentang nilai 0 sampai 1.

b. Implementasi Pembentukan Bias

Implementasi bias dilakukan untuk memperoleh nilai bias secara acak dengan rentang nilai 0 sampai dengan 1. Ukuran bias ditentukan dengan rumus pada Persamaan 2. Berikut implementasi pembentukan bias pada Kode Program 5.4.

Kode Program 5.4 Implementasi Pembentukan Bias

```
1.  // method load bias
2.          private void LoadBias()
3.          {
4.              if (CheckData())
5.              {
6.                  //untuk judul kolom di view
7.                  var headerNames = new List<object> {"Bias"};
8.                  var datas = new double[DataHelper.Current.HiddenLayer,
9.  1];
10.                 var random = new Random();
11.                 //random array untuk bobot dg range 0-1
12.                 for (var i = 0; i < DataHelper.Current.HiddenLayer;
13. i++)
14.                 {
15.                     for (int j = 0; j < 1; j++)
16.                     {
17.                         datas[i, j] = random.NextDouble();
18.                     }
19.                 }
20.
21.                 BiasDatas = new ArrayDataView(datas,
22. headerNames.ToArray());
```

```
22.                 DataHelper.Current.BiasDatas = BiasDatas; // data
23. disimpan dalam variabel
24.             }
25.         }
```

Penjelasan Kode Program 5.4 tentang implementasi pembentukan biasadalah sebagai berikut:

Baris 1 s.d 3	:	Method pembentukan bias.
Baris 4 s.d 5	:	Cek ketersediaan data.
Baris 6 s.d 7	:	Pemberian nama pada *header* kolom pada bias.
Baris 8 s.d 9	:	Inisialisasi variabel array bias.
Baris 10	:	Inisialisasi variabel random untuk nilai acak antara 0 sampai 1.
Baris 11 s.d 20	:	Menghitung nilai bias secara random dengan rentang nilai 0 sampai dengan 1.
Baris 21 s.d 25	:	Menyimpan nilai bias kedalam variabel yang diperoleh secara acak dengan rentang nilai 0 sampai 1.

c. Implementasi Solusi Penyelesaian *Extreme Learning Machine*

Implementasi solusi penyelesaian *extreme learning machine* dilakukan untuk memperoleh keluaran berupa pengelompokkan kelas yang dievaluasi berdasarkan tingkat akurasinya. Berikut implementasi solusi penyelesaian ELM pada Kode Program 5.5.

Kode Program 5.5 Implementasi Solusi Penyelesaian ELM

```
1.  private void CountFold(IReadOnlyList<double[,]> inputs, double[,]
2.  testing, int testingFold)
3.          {
4.              int dataBeratCount;
5.              double[,] input1 = {}, input2 = {}, input3 = {}, input4 =
6.  {};
7.              int rowLengthInput2 = 0, rowLengthInput3 = 0,
8.  rowLengthInput4 = 0;
9.              for (var i = 0; i < inputs.Count; i++)
10.             {
11.                 if (i == 0)
12.                 {
13.                     input1 = inputs[i];
14.                 }
15.                 if (i == 1)
16.                 {
17.                     input2 = inputs[i];
18.                 }
19.                 if (i == 2)
20.                 {
21.                     input3 = inputs[i];
22.                 }
22.                 if (i == 3)
23.                 {
24.                     input4 = inputs[i];
25.                 }
26.             }
```

```
            var rowLengthInput1 = input1.Length/ColumnInNormalisasi;
            var totalLength = rowLengthInput1;
            if (DataHelper.Current.KValue > 2)
            {
                rowLengthInput2 = input2.Length/ColumnInNormalisasi;
                totalLength += rowLengthInput2;
                if (DataHelper.Current.KValue > 3)
                {
                    rowLengthInput3 =
input3.Length/ColumnInNormalisasi;
                    totalLength += rowLengthInput3;
                    if (DataHelper.Current.KValue > 4)
                    {
                        rowLengthInput4 =
input4.Length/ColumnInNormalisasi;
                        totalLength += rowLengthInput4;
                    }
                }
            }
            var newData = new double[totalLength,
ColumnInNormalisasi];
            if (testingFold == 5)
            {
                dataBeratCount =
DataHelper.Current.Fold5Datas.Count(x => x.KlasWeight == 3);
            }
            else if (testingFold == 4)
            {
                dataBeratCount =
DataHelper.Current.Fold4Datas.Count(x => x.KlasWeight == 3);
            }
            else if (testingFold == 3)
            {
                dataBeratCount =
DataHelper.Current.Fold3Datas.Count(x => x.KlasWeight == 3);
            }
            else if (testingFold == 2)
            {
                dataBeratCount =
DataHelper.Current.Fold2Datas.Count(x => x.KlasWeight == 3);
            }
            else
            {
                dataBeratCount =
DataHelper.Current.Fold1Datas.Count(x => x.KlasWeight == 3);
            }
            for (var i = 0; i < totalLength; i++)
            {
                for (var j = 0; j < ColumnInNormalisasi; j++)
                {
                    if (DataHelper.Current.KValue == 2)
                    {
                        newData[i, j] = input1[i, j];
                    }
                    else if (DataHelper.Current.KValue == 3)
                    {
                        if (i < rowLengthInput1)
                        {
                            newData[i, j] = input1[i, j];
                        }
```

```
                    else
                    {
                        newData[i, j] = input2[i -
rowLengthInput1, j];
                    }
                }
                else if (DataHelper.Current.KValue == 4)
                {
                    if (i < rowLengthInput1)
                    {
                        newData[i, j] = input1[i, j];
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2)
                    {
                        newData[i, j] = input2[i -
rowLengthInput1, j];
                    }
                    else
                    {
                        newData[i, j] = input3[i -
(rowLengthInput1 + rowLengthInput2), j];
                    }
                }
                else if (DataHelper.Current.KValue == 5)
                {
                    if (i < rowLengthInput1)
                    {
                        newData[i, j] = input1[i, j];
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2)
                    {
                        newData[i, j] = input2[i -
rowLengthInput1, j];
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2 + rowLengthInput3)
                    {
                        newData[i, j] = input3[i -
(rowLengthInput1 + rowLengthInput2), j];
                    }
                    else
                    {
                        newData[i, j] = input4[i -
(rowLengthInput1 + rowLengthInput2 + rowLengthInput3), j];
                    }
                }
            }
        }
        var newMatrixBuilder = Matrix<double>.Build;
        var newMatrix = newMatrixBuilder.DenseOfArray(newData);
        //-------Fold 1--------

        // inisialisasi variabel Fold 1 dikalikan Transpose Bobot
        var resultFold1MultipyTransposeBobot =
newMatrix.Multiply(DataHelper.Current.TransposedBobot);

        // nilai Fold 1 dijumlahkan dengan Bias
        var resultAddBiasFold1 =
```

```
                new double[resultFold1MultipyTransposeBobot.RowCount,
resultFold1MultipyTransposeBobot.ColumnCount];

            // penjumlahan dengan bias matriks hasil perkalian Fold 1
dengan Tranpose Bobot
            for (var i = 0; i <
resultFold1MultipyTransposeBobot.RowCount; i++)
            {
                for (var j = 0; j <
resultFold1MultipyTransposeBobot.ColumnCount; j++)
                {
                    // menjumlahkan kolom bias dengan kolom matriks
yang sama secara berurutan
                    // fungsi keluaran hidden layer
                    resultAddBiasFold1[i, j] =
resultFold1MultipyTransposeBobot[i, j] + DataHelper.Current.Bias[j,
0];
                }
            }

            // menghitung nilai fungsi aktivasi sigmoid biner dari
fungsi keluaran hidden layer
            for (var i = 0; i <
resultFold1MultipyTransposeBobot.RowCount; i++)
            {
                for (var j = 0; j <
resultFold1MultipyTransposeBobot.ColumnCount; j++)
                {
                    resultAddBiasFold1[i, j] = 1/(1 + Math.Exp(-
resultAddBiasFold1[i, j]));
                }
            }

            // membangun matriks fungsi aktivasi Fold 1
            var matrixBuilderAfterExponenFold1 =
Matrix<double>.Build;
            var matrixAfterExponenFold1 =
matrixBuilderAfterExponenFold1.DenseOfArray(resultAddBiasFold1);

            // menghitung transpose matriks
            var transposedMatrixWithExponenFold1 =
matrixAfterExponenFold1.Transpose();

            // operasi perkalian matriks fungsi dengan transpose
matriksnya
            var resultTransposeExponenMultiplicationFold1 =

transposedMatrixWithExponenFold1.Multiply(matrixAfterExponenFold1);

            // hasilnya di inverse
            var inversedMatrixFold1 =
resultTransposeExponenMultiplicationFold1.Inverse();

            // diperoleh nilai matriks moore-penrose
            var matrixMoorpenRooseFold1 =
inversedMatrixFold1.Multiply(transposedMatrixWithExponenFold1);
            var matrixBuilderKlasFold1 = Matrix<double>.Build;

            // memasukkan nilai target ke dalam array
            var arrayKlasFold1 = new double[totalLength, 1];
```

```
207.            if (DataHelper.Current.KValue == 2)
208.            {
209.                if (testingFold == 2)
210.                {
211.                    for (var i = 0; i < arrayKlasFold1.Length; i++)
212.                    {
213.                        var fold1Data =
214. DataHelper.Current.Fold1Datas[i];
215.                        arrayKlasFold1[i, 0] = fold1Data.KlasWeight;
216.                    }
217.                }
218.                else
219.                {
220.                    for (var i = 0; i < arrayKlasFold1.Length; i++)
221.                    {
222.                        var fold1Data =
223. DataHelper.Current.Fold2Datas[i];
224.                        arrayKlasFold1[i, 0] = fold1Data.KlasWeight;
225.                    }
226.                }
227.            }
228.            else if (DataHelper.Current.KValue == 3)
229.            {
230.                if (testingFold == 3)
231.                {
232.                    for (var i = 0; i < arrayKlasFold1.Length; i++)
233.                    {
234.                        if (i < rowLengthInput1)
235.                        {
236.                            var fold1Data =
237. DataHelper.Current.Fold1Datas[i];
238.                            arrayKlasFold1[i, 0] =
239. fold1Data.KlasWeight;
240.                        }
241.                        else
242.                        {
243.                            var fold2Data =
244. DataHelper.Current.Fold2Datas[i - rowLengthInput1];
245.                            arrayKlasFold1[i, 0] =
246. fold2Data.KlasWeight;
247.                        }
248.                    }
249.                }
250.                else if (testingFold == 2)
251.                {
252.                    for (var i = 0; i < arrayKlasFold1.Length; i++)
253.                    {
254.                        if (i < rowLengthInput1)
255.                        {
256.                            var fold1Data =
257. DataHelper.Current.Fold1Datas[i];
258.                            arrayKlasFold1[i, 0] =
259. fold1Data.KlasWeight;
260.                        }
261.                        else
262.                        {
263.                            var fold2Data =
264. DataHelper.Current.Fold3Datas[i - rowLengthInput1];
265.                            arrayKlasFold1[i, 0] =
266. fold2Data.KlasWeight;
```

```
                    }
                }
            }
            else
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput1)
                    {
                        var fold1Data =
DataHelper.Current.Fold2Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else
                    {
                        var fold2Data =
DataHelper.Current.Fold3Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                }
            }
        }
        else if (DataHelper.Current.KValue == 4)
        {
            if (testingFold == 4)
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput1)
                    {
                        var fold1Data =
DataHelper.Current.Fold1Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2)
                    {
                        var fold2Data =
DataHelper.Current.Fold2Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                    else
                    {
                        var fold3Data =
DataHelper.Current.Fold3Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                }
            }
            else if (testingFold == 3)
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput1)
```

```
                    {
                        var fold1Data =
DataHelper.Current.Fold1Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2)
                    {
                        var fold2Data =
DataHelper.Current.Fold2Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                    else
                    {
                        var fold3Data =
DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                }
            }
            else if (testingFold == 2)
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput1)
                    {
                        var fold1Data =
DataHelper.Current.Fold1Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput2)
                    {
                        var fold2Data =
DataHelper.Current.Fold3Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                    else
                    {
                        var fold3Data =
DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                }
            }
            else
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput2)
                    {
                        var fold1Data =
```

```
388.  DataHelper.Current.Fold2Datas[i];
389.                              arrayKlasFold1[i, 0] =
390.  fold1Data.KlasWeight;
391.                          }
392.                          else if (i < rowLengthInput2 +
393.  rowLengthInput1)
394.                          {
395.                              var fold2Data =
396.  DataHelper.Current.Fold3Datas[i - rowLengthInput2];
397.                              arrayKlasFold1[i, 0] =
398.  fold2Data.KlasWeight;
399.                          }
400.                          else
401.                          {
402.                              var fold3Data =
403.  DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
404.  rowLengthInput2)];
405.                              arrayKlasFold1[i, 0] =
406.  fold3Data.KlasWeight;
407.                          }
408.                      }
409.                  }
410.              }
411.              else if (DataHelper.Current.KValue == 5)
412.              {
413.                  if (testingFold == 5)
414.                  {
415.                      for (var i = 0; i < arrayKlasFold1.Length; i++)
416.                      {
417.                          if (i < rowLengthInput1)
418.                          {
419.                              var fold1Data =
420.  DataHelper.Current.Fold1Datas[i];
421.                              arrayKlasFold1[i, 0] =
422.  fold1Data.KlasWeight;
423.                          }
424.                          else if (i < rowLengthInput1 +
425.  rowLengthInput2)
426.                          {
427.                              var fold2Data =
428.  DataHelper.Current.Fold2Datas[i - rowLengthInput1];
429.                              arrayKlasFold1[i, 0] =
430.  fold2Data.KlasWeight;
431.                          }
432.                          else if (i < rowLengthInput1 +
433.  rowLengthInput2 + rowLengthInput3)
434.                          {
435.                              var fold3Data =
436.  DataHelper.Current.Fold3Datas[i - (rowLengthInput1 +
437.  rowLengthInput2)];
438.                              arrayKlasFold1[i, 0] =
439.  fold3Data.KlasWeight;
440.                          }
441.                          else
442.                          {
443.                              var fold4Data =
444.                                  DataHelper.Current.Fold4Datas[i -
445.  (rowLengthInput1 + rowLengthInput2 + rowLengthInput3)];
446.                              arrayKlasFold1[i, 0] =
447.  fold4Data.KlasWeight;
```

```
448.                         }
449.                     }
450.                 }
451.                 else if (testingFold == 4)
452.                 {
453.                     for (var i = 0; i < arrayKlasFold1.Length; i++)
454.                     {
455.                         if (i < rowLengthInput1)
456.                         {
457.                             var fold1Data =
458. DataHelper.Current.Fold1Datas[i];
459.                             arrayKlasFold1[i, 0] =
460. fold1Data.KlasWeight;
461.                         }
462.                         else if (i < rowLengthInput1 +
463. rowLengthInput2)
464.                         {
465.                             var fold2Data =
467. DataHelper.Current.Fold2Datas[i - rowLengthInput1];
468.                             arrayKlasFold1[i, 0] =
469. fold2Data.KlasWeight;
470.                         }
471.                         else if (i < rowLengthInput1 +
472. rowLengthInput2 + rowLengthInput3)
473.                         {
474.                             var fold3Data =
475. DataHelper.Current.Fold3Datas[i - (rowLengthInput1 +
476. rowLengthInput2)];
477.                             arrayKlasFold1[i, 0] =
478. fold3Data.KlasWeight;
479.                         }
450.                         else
451.                         {
452.                             var fold4Data =
453.                                 DataHelper.Current.Fold5Datas[i -
454. (rowLengthInput1 + rowLengthInput2 + rowLengthInput3)];
455.                             arrayKlasFold1[i, 0] =
456. fold4Data.KlasWeight;
457.                         }
458.                     }
459.                 }
460.                 else if (testingFold == 3)
461.                 {
462.                     for (var i = 0; i < arrayKlasFold1.Length; i++)
463.                     {
464.                         if (i < rowLengthInput1)
465.                         {
466.                             var fold1Data =
467. DataHelper.Current.Fold1Datas[i];
468.                             arrayKlasFold1[i, 0] =
469. fold1Data.KlasWeight;
470.                         }
471.                         else if (i < rowLengthInput1 +
472. rowLengthInput2)
473.                         {
474.                             var fold2Data =
475. DataHelper.Current.Fold2Datas[i - rowLengthInput1];
476.                             arrayKlasFold1[i, 0] =
477. fold2Data.KlasWeight;
478.                         }
```

```
                    else if (i < rowLengthInput1 +
rowLengthInput2 + rowLengthInput4)
                    {
                        var fold3Data =
DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                    else
                    {
                        var fold4Data =
                            DataHelper.Current.Fold5Datas[i -
(rowLengthInput1 + rowLengthInput2 + rowLengthInput3)];
                        arrayKlasFold1[i, 0] =
fold4Data.KlasWeight;
                    }
                }
            }
            else if (testingFold == 2)
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
                {
                    if (i < rowLengthInput1)
                    {
                        var fold1Data =
DataHelper.Current.Fold1Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput3)
                    {
                        var fold2Data =
DataHelper.Current.Fold3Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                    else if (i < rowLengthInput1 +
rowLengthInput3 + rowLengthInput4)
                    {
                        var fold3Data =
DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                    else
                    {
                        var fold4Data =
                            DataHelper.Current.Fold5Datas[i -
(rowLengthInput1 + rowLengthInput2 + rowLengthInput3)];
                        arrayKlasFold1[i, 0] =
fold4Data.KlasWeight;
                    }
                }
            }
            else
            {
                for (var i = 0; i < arrayKlasFold1.Length; i++)
```

```
                {
                    if (i < rowLengthInput2)
                    {
                        var fold1Data =
DataHelper.Current.Fold2Datas[i];
                        arrayKlasFold1[i, 0] =
fold1Data.KlasWeight;
                    }
                    else if (i < rowLengthInput2 +
rowLengthInput3)
                    {
                        var fold2Data =
DataHelper.Current.Fold3Datas[i - rowLengthInput1];
                        arrayKlasFold1[i, 0] =
fold2Data.KlasWeight;
                    }
                    else if (i < rowLengthInput2 +
rowLengthInput3 + rowLengthInput4)
                    {
                        var fold3Data =
DataHelper.Current.Fold4Datas[i - (rowLengthInput1 +
rowLengthInput2)];
                        arrayKlasFold1[i, 0] =
fold3Data.KlasWeight;
                    }
                    else
                    {
                        var fold4Data =
                            DataHelper.Current.Fold5Datas[i -
(rowLengthInput1 + rowLengthInput2 + rowLengthInput3)];
                        arrayKlasFold1[i, 0] =
fold4Data.KlasWeight;
                    }
                }
            }
        }

        // membangun matriks target
        var matrixKlasFold1 =
matrixBuilderKlasFold1.DenseOfArray(arrayKlasFold1);

        // hasil keluaran dari proses training Fold 1 berupa
bobot output
        var resultMatrixInput =
matrixMoorpenRooseFold1.Multiply(matrixKlasFold1);

        var testingMatrixBuilder = Matrix<double>.Build;
        var testingMatrix =
testingMatrixBuilder.DenseOfArray(testing);
        //Fold 3
        var resultFold3MultipyTransposeBobot =
testingMatrix.Multiply(DataHelper.Current.TransposedBobot);
        var resultAddBiasFold3 =
            new double[resultFold3MultipyTransposeBobot.RowCount,
resultFold3MultipyTransposeBobot.ColumnCount];
        for (var i = 0; i <
resultFold3MultipyTransposeBobot.RowCount; i++)
        {
            for (var j = 0; j <
resultFold3MultipyTransposeBobot.ColumnCount; j++)
```

```
                {
                    resultAddBiasFold3[i, j] =
resultFold3MultipyTransposeBobot[i, j] + DataHelper.Current.Bias[j,
0];
                }
            }

            for (var i = 0; i <
resultFold3MultipyTransposeBobot.RowCount; i++)
            {
                for (var j = 0; j <
resultFold3MultipyTransposeBobot.ColumnCount; j++)
                {
                    resultAddBiasFold3[i, j] = 1/(1 + Math.Exp(-
resultAddBiasFold3[i, j]));
                }
            }
            var matrixBuilderAfterExponenFold3 =
Matrix<double>.Build;
            var matrixAfterExponenFold3 =
matrixBuilderAfterExponenFold3.DenseOfArray(resultAddBiasFold3);
            var result1Fold3 =
matrixAfterExponenFold3.Multiply(resultMatrixInput);
            //var result2Fold3 =
matrixAfterExponenFold3.Multiply(resultFold2);

            var arrayResultFold1 = new double[result1Fold3.RowCount,
7];
            var dataNotSameCount = 0;

            for (var i = 0; i < result1Fold3.RowCount; i++)
            {
                for (var j = 0; j < result1Fold3.ColumnCount; j++)
                {
                    arrayResultFold1[i, j] = result1Fold3[i, j];
                    arrayResultFold1[i, j + 1] =
Math.Abs(result1Fold3[i, j] - 1);
                    arrayResultFold1[i, j + 2] =
Math.Abs(result1Fold3[i, j] - 2);
                    arrayResultFold1[i, j + 3] =
Math.Abs(result1Fold3[i, j] - 3);
                    var arrayValues = new[]
                    {
                        arrayResultFold1[i, j + 1],
arrayResultFold1[i, j + 2],
                        arrayResultFold1[i, j + 3]
                    };
                    arrayResultFold1[i, j + 4] = arrayValues.Min();
                    arrayResultFold1[i, j + 5] =
Array.IndexOf(arrayValues, arrayValues.Min()) + 1;
                    if (i < DataHelper.Current.RinganDataCount)
                    {
                        arrayResultFold1[i, j + 6] = 1;
                    }
                    else if (i < result1Fold3.RowCount -
dataBeratCount)
                    {
                        arrayResultFold1[i, j + 6] = 2;
                    }
                    else
```

```
                    {
                        arrayResultFold1[i, j + 6] = 3;
                    }
                    if ((int) arrayResultFold1[i, j + 5] != (int)
arrayResultFold1[i, j + 6])
                    {
                        dataNotSameCount++;
                    }
                }
            }
            var dataCount = result1Fold3.RowCount;
            var akurasi = (double) (dataCount -
dataNotSameCount)/dataCount*100;
            if (testingFold == 5)
            {
                SolusiDataViewFold5 = new ArrayDataView(
                    arrayResultFold1,
                    new object[] {"Output", "Klas 1", "Klas 2", "Klas
3", "Nilai Minimal", "Prediksi", "Klas"});
                AkurasiFold5 = akurasi + "%";
            }
            if (testingFold == 4)
            {
                SolusiDataViewFold4 = new ArrayDataView(
                    arrayResultFold1,
                    new object[] {"Output", "Klas 1", "Klas 2", "Klas
3", "Nilai Minimal", "Prediksi", "Klas"});
                AkurasiFold4 = akurasi + "%";
            }
            if (testingFold == 3)
            {
                SolusiDataViewFold3 = new ArrayDataView(
                    arrayResultFold1,
                    new object[] {"Output", "Klas 1", "Klas 2", "Klas
3", "Nilai Minimal", "Prediksi", "Klas"});
                AkurasiFold3 = akurasi + "%";
            }
            else if (testingFold == 2)
            {
                SolusiDataViewFold2 = new ArrayDataView(
                    arrayResultFold1,
                    new object[] {"Output", "Klas 1", "Klas 2", "Klas
3", "Nilai Minimal", "Prediksi", "Klas"});
                AkurasiFold2 = akurasi + "%";
            }
            else
            {
                SolusiDataViewFold1 = new ArrayDataView(
                    arrayResultFold1,
                    new object[] {"Output", "Klas 1", "Klas 2", "Klas
3", "Nilai Minimal", "Prediksi", "Klas"});
                AkurasiFold1 = akurasi + "%";
            }
        }
```

Penjelasan Kode Program 5.5 tentang implementasi solusi penyelesaian ELM adalah sebagai berikut:

Baris 1 s.d 3	:	Method solusi penyelesaian ELM.
Baris 4 s.d 26	:	Pembentukan baris untuk data *training* secara bergantian berdasarkan nilai *k* yang dimasukkan.
Baris 27 s.d 45	:	Pembentukan kolom untuk data*training* secara bergantian berdasarkan nilai *k* yang dimasukkan.
Baris 46 s.d 69	:	Inisialisasi variabel untuk pengambilan data *testing* untuk diletakkan pada setiap *fold* secara bergantian.
Baris 70 s.d 138	:	Proses pengambilan data *training* dan diletakkan pada semua *fold* secara bergantian berdasarkan pembagiannya sesuai nilai k yang dimasukkan.
Baris 139 s.d 144	:	Proses perkalian untuk data *training* dengan bobot awal.
Baris 145 s.d 166	:	Proses menjumlahkan hasil perkalian dengan bias berdasarkan kolom secara berurutan.
Baris 167 s.d 185	:	Proses aktivasi fungsi keluaran *hidden layer* yang diperoleh dari proses sebelumnya yaitu pada Baris 139 sampai 166.
Baris 186 s.d 204	:	Proses menghitung matriks *moore-penrose pseudo inverse*.
Baris 205 s.d 582	:	Proses pembentukan matriks target yang diambil dari kelas pada setiap *fold*.
Baris 583 s.d 587	:	Hasil keluaran proses *training* berupa nilai bobot *output*.
Baris 588 s.d 627	:	Proses *testing* menghasilkan keluaran hasil perkalian fungsi keluaran *hidden layer* dengan bobot *output* dari proses *training*.
Baris 628 s.d 671	:	Masuk proses evaluasi penentuan. Proses pada baris ini menghitung jarak antara keluaran dengan masing – masing kelas.
Baris 672 s.d 715	:	Proses menghitung akurasi.

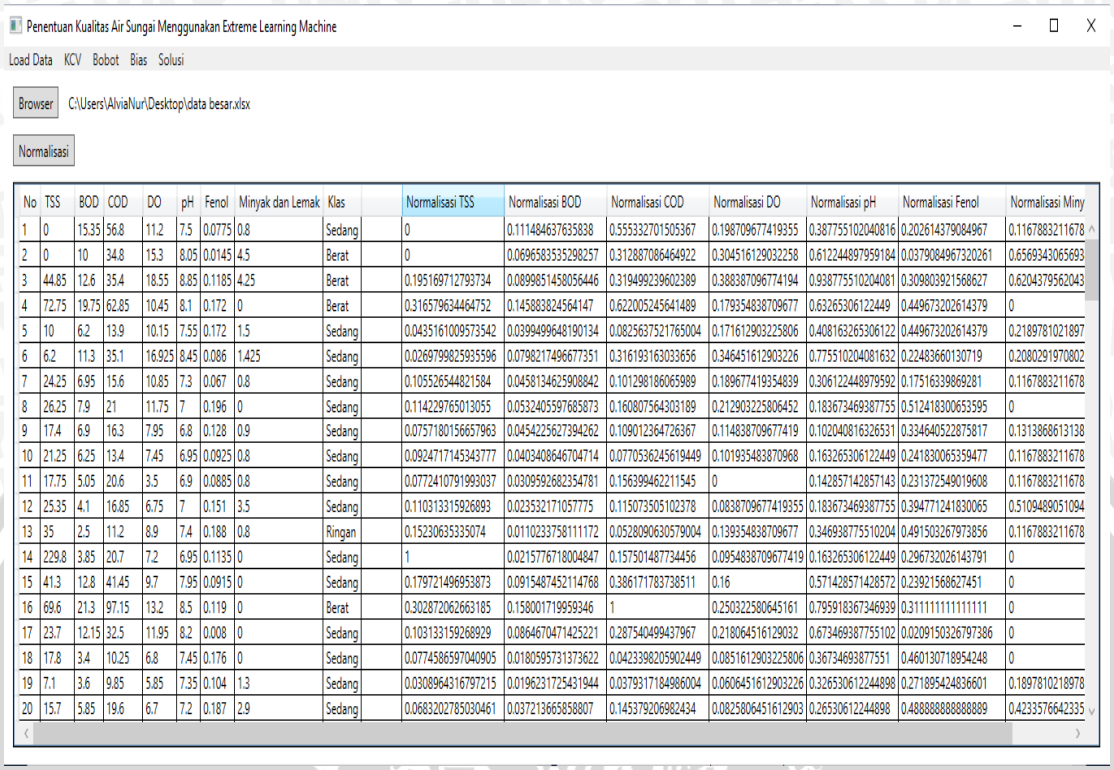
5.2 Implementasi Antarmuka

Implementasi antarmuka merupakan hasil perancangan antarmuka yang telah dibuat sebelumnya. Antarmuka penentuan kulitas air sungai menggunakan *extreme learning machine* terdiri dari 5 antarmuka utama meliputi antarmuka *load* data, antarmuka *k-fold cross validation*, antarmuka bobot, antarmuka bias, dan antarmuka solusi.

5.2.1 Implementasi Antarmuka *Load* Data

Implementasi antarmuka*load*data merupakan antarmuka utama pada sistem ini. Antarmuka ini berfungsi untuk menerima masukan *dataset* kualitas air

sungai. Antarmuka ini juga berfungsi untuk me-*load* data dan normalisasi. Hasilnya akan ditampilkan pada *datagridview* dalam bentuk tabel. Implementasi antarmuka *load* data ditunjukkan pada Gambar 5.1 sebagai berikut:



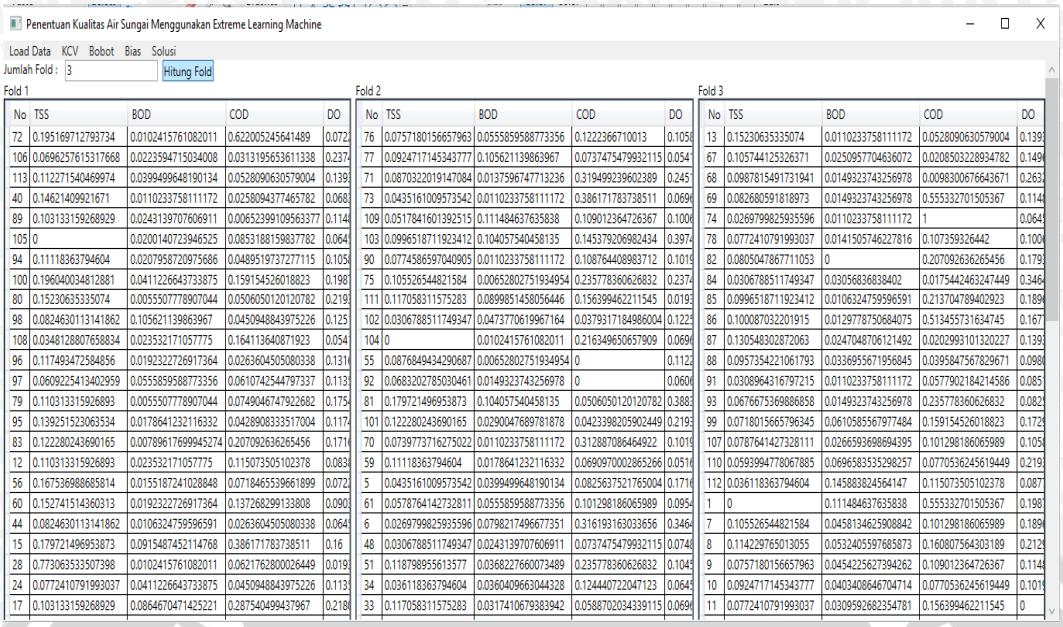
No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Klas	Normalisasi TSS	Normalisasi BOD	Normalisasi COD	Normalisasi DO	Normalisasi pH	Normalisasi Fenol	Normalisasi Miny
1	0	15.35	56.8	11.2	7.5	0.0775	0.8	Sedang	0	0.111484637635838	0.555332701505367	0.198709677419355	0.387755102040816	0.202614379084967	0.1167883211678
2	0	10	34.8	15.3	8.05	0.0145	4.5	Berat	0	0.0696583535298257	0.312887086464922	0.304516129032258	0.612244897959184	0.0379084967320261	0.6569343065693
3	44.85	12.6	35.4	18.55	8.85	0.1185	4.25	Berat	0.195169712793734	0.0899851458056446	0.319499239602389	0.388387096774194	0.938775510204081	0.309803921568627	0.6204379562043
4	72.75	19.75	62.85	10.45	8.1	0.172	0	Berat	0.316579634464752	0.145883824564147	0.622005245641489	0.179354838709677	0.63265306122449	0.449673202614379	0
5	10	6.2	13.9	10.15	7.55	0.172	1.5	Sedang	0.0435161009573542	0.0399499648190134	0.0825637521765004	0.171612903225806	0.408163265306122	0.449673202614379	0.2189781021897
6	6.2	11.3	35.1	16.925	8.45	0.086	1.425	Sedang	0.0269799825935596	0.0798217496677351	0.316193163033656	0.346451612903226	0.775510204081632	0.224836601307191	0.2080291970802
7	24.25	6.95	15.6	10.85	7.3	0.067	0.8	Sedang	0.105526544821584	0.0458134625908842	0.101298186065989	0.189677419354839	0.306122448979592	0.175163398692810	0.1167883211678
8	26.25	7.9	21	11.75	7	0.196	0	Sedang	0.114229765013055	0.0532405597685873	0.160807564303189	0.212903225806452	0.183673469387755	0.512418300653595	0
9	17.4	6.9	16.3	7.95	6.8	0.128	0.9	Sedang	0.0757180156657963	0.0454225627394262	0.109012364726367	0.114838709677419	0.102040816326531	0.334640522875817	0.1313868613138
10	21.25	6.25	13.4	7.45	6.95	0.0925	0.8	Sedang	0.0924717145343777	0.0403408646704714	0.0770536245619449	0.101935483870968	0.163265306122449	0.241830065359477	0.1167883211678
11	17.75	5.05	20.6	3.5	6.9	0.0885	0.8	Sedang	0.0772410791993037	0.0309592682354781	0.156399462211545	0	0.142857142857143	0.231372549019608	0.1167883211678
12	25.35	4.1	16.85	6.75	7	0.151	3.5	Sedang	0.110313319926893	0.023532171057775	0.115073505102378	0.0838709677419355	0.163673469387755	0.394771241830065	0.5109489051094
13	35	2.5	11.2	8.9	7.4	0.188	0.8	Ringan	0.15230635335074	0.0110233758111172	0.0528090630579004	0.139354838709677	0.346938775510204	0.491503267973856	0.1167883211678
14	229.8	3.85	20.7	7.2	6.95	0.1135	0	Sedang	1	0.0215776718004847	0.157501487734456	0.0954838709677419	0.163265306122449	0.296732026143791	0
15	41.3	12.8	41.45	9.7	7.95	0.0915	0	Sedang	0.179721496953873	0.0915487452114768	0.386171783738511	0.16	0.571428571428572	0.23921568627451	0
16	69.6	21.3	97.15	13.2	8.5	0.119	0	Berat	0.302872062663185	0.158001719959346	1	0.250322580645161	0.795918367346939	0.311111111111111	0
17	23.7	12.15	32.5	11.95	8.2	0.008	0	Sedang	0.103133159268929	0.0864670471425221	0.287540499437967	0.218064516129032	0.673469387755102	0.0209150326797386	0
18	17.8	3.4	10.25	6.8	7.45	0.176	0	Sedang	0.0774586597040905	0.0180595731373622	0.0423398205902449	0.0851612903225806	0.367346938775510	0.460130718954248	0
19	7.1	3.6	9.85	5.85	7.35	0.104	1.3	Sedang	0.0308964316797215	0.0196231725431944	0.0379317184986004	0.0606451612903226	0.326530612244898	0.271895424836601	0.1897810218978
20	15.7	5.85	19.6	6.7	7.2	0.187	2.9	Sedang	0.0683202785030461	0.037213665858807	0.145379206982434	0.0825806451612903	0.265306122448980	0.488888888888889	0.4233576642335

Gambar 5.1 Implementasi Antarmuka *Load* Data

Gambar 5.1 menjelaskan hasil implementasi antarmuka*load* data. Proses *load* data diawali dengan pengguna menekan tombol *browse file* bertujuan untuk memasukkan *file* berisi data kualitas air sungai dengan format .xls dari komputer pengguna. Lokasi *file* ditampilkan pada *textbox* pertama. Selanjutnya, data dari *file* diterima oleh sistem disimpanke dalam *array* (kebutuhan komputasi). Kemudian, pengguna menekan tombol normalisai untuk melakukan proses normalisasi. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview* pada sisi kiri merupakan *dataset* dan sisi kanan merupakan data yang telah dinormalisasi.

5.2.2 Implementasi Antarmuka K-*Fold Cross Validation*

Implementasi antarmuka *k-fold cross validation* merupakan antarmuka kedua pada sistem ini. Antarmuka ini berfungsi untuk membagi *dataset* yang telah dimasukkan oleh pengguna. Data dibagi dalam bentuk *fold* yang jumlahnya berdasarkan nilai *k* yang diinputkan oleh pengguna. Pembagian data pada masing *fold* menggunakan Persamaan 10 secara acak dengan ketentuan data yang sudah dilakukan pengambilan tidak boleh dilakukan pengambilan kembali pada *fold* lain dan sisa bagi data akan diletakkan pada *fold* terakhir. Data ditampilkan pada *datagridview* pada masing – masing *fold*. Implementasi antarmuka *k-fold cross validation* ditunjukkan pada Gambar 5.2 sebagai berikut:

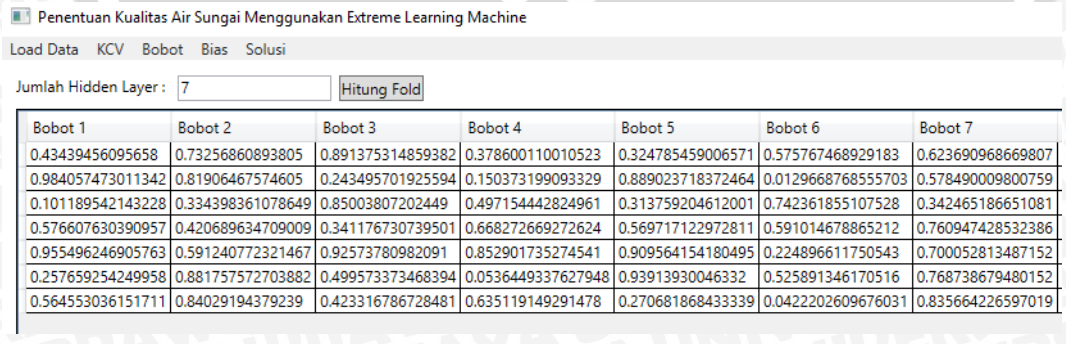


Gambar 5.2 Implementasi Antarmuka K-*Fold Cross Validation*

Gambar 5.2 menjelaskan hasil implementasi antarmuka *k-fold cross validation*. Proses membagi data diawali dengan pengguna memasukkan nilai *k* antara 2 sampai 5. Kemudian, pengguna menekan tombol Hitung Fold untuk melakukan proses pembagian data berupa *fold* sesuai nilai masukan pengguna. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview* pada masing – masing *fold*. Data pada setiap *fold* berbeda dengan *fold* yang lain dan sisa bagi data diletakkan pada *fold* terakhir.

5.2.3 Implementasi Antarmuka Bobot

Implementasi antarmuka bobot merupakan antarmuka ketiga pada sistem ini. Antarmuka ini berfungsi untuk menerima masukan jumlah *hidden node* antara 2 sampai 7. Antarmuka ini juga berfungsi untuk membangun bobot awal menggunakan Persamaan 1. Hasilnya akan ditampilkan pada *datagridview* dalam bentuk tabel. Implementasi antarmuka bobot ditunjukkan pada Gambar 5.3 sebagai berikut:



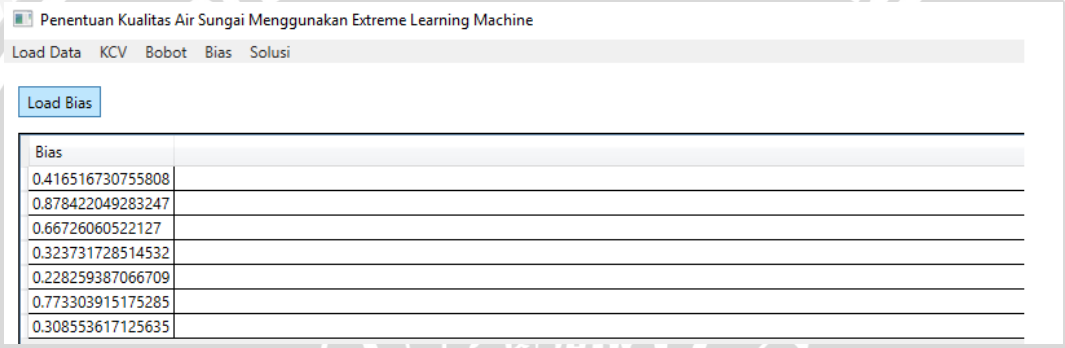
Bobot 1	Bobot 2	Bobot 3	Bobot 4	Bobot 5	Bobot 6	Bobot 7
0.43439456095658	0.73256860893805	0.891375314859382	0.378600110010523	0.324785459006571	0.575767468929183	0.623690968669807
0.984057473011342	0.81906467574605	0.243495701925594	0.150373199093329	0.889023718372464	0.0129668768555703	0.578490009800759
0.101189542143228	0.334398361078649	0.85003807202449	0.497154442824961	0.313759204612001	0.742361855107528	0.342465186651081
0.576607630390957	0.420689634709009	0.341176730739501	0.668272669272624	0.569717122972811	0.591014678865212	0.760947428532386
0.955496246905763	0.591240772321467	0.92573780982091	0.852901735274541	0.909564154180495	0.224896611750543	0.700052813487152
0.257659254249958	0.881757572703882	0.499573373468394	0.0536449337627948	0.939139300046332	0.525891346170516	0.768738679480152
0.564553036151711	0.84029194379239	0.423316786728481	0.635119149291478	0.270681868433339	0.0422202609676031	0.835664226597019

Gambar 5.3 Implementasi Antarmuka Bobot

Gambar 5.3 menjelaskan hasil implementasi antarmuka bobot awal. Proses membangun bobot awal menggunakan Persamaan 1 dengan inputan jumlah *hidden node* antara 2 sampai 7 oleh pengguna. Kemudian, pengguna menekan tombol Hitung untuk melakukan proses perhitungan bobot awal secara *random* dengan interval 0 sampai 1 sesuai nilai *hidden node* masukan pengguna. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview*.

5.2.4 Implementasi Antarmuka Bias

Implementasi antarmuka bias merupakan antarmuka keempat pada sistem ini. Antarmuka ini berfungsi untuk membangun bias masukan jumlah *hidden layer* antara 2 sampai 7 pada antarmuka bobo sebelumnya. Antarmuka ini juga berfungsi untuk membangun bias menggunakan Persamaan 2. Hasilnya akan ditampilkan pada *datagridview* dalam bentuk tabel. Implementasi antarmuka bias ditunjukkan pada Gambar 5.4 sebagai berikut:

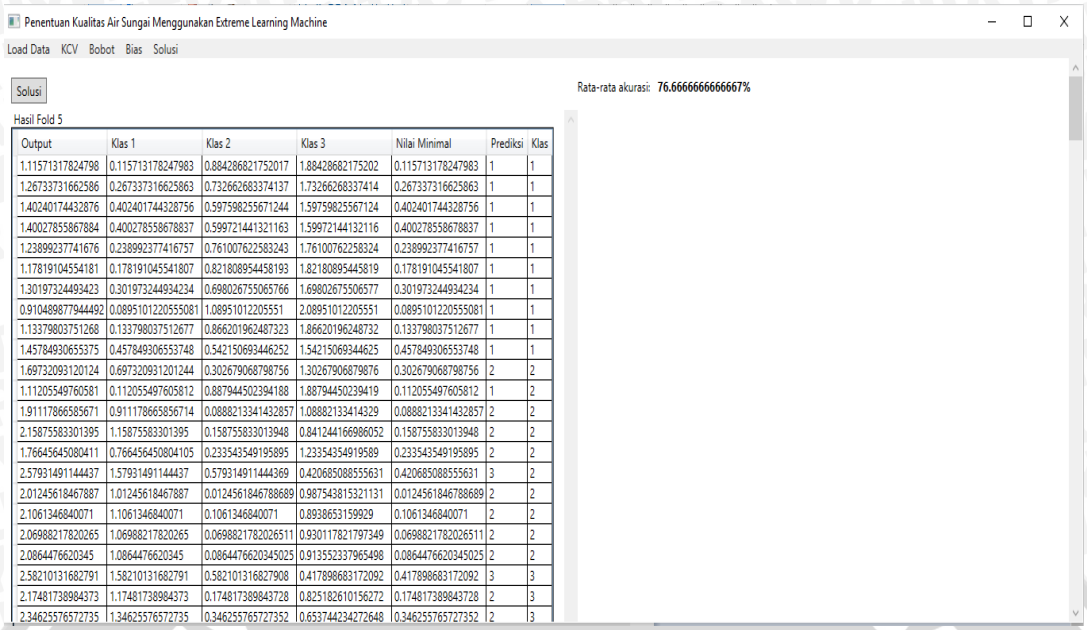


Gambar 5.4 Implementasi Antarmuka Bias

Gambar 5.4 menjelaskan hasil implementasi antarmuka bias. Proses membangun bias menggunakan Persamaan 2 dengan inputan jumlah *hidden layer* antara 2 sampai 7 oleh pengguna pada antarmuka bobot sebelumnya. Kemudian, pengguna menekan tombol Load Bias untuk melakukan proses perhitungan bias secara *random* dengan interval 0 sampai 1 sesuai nilai *hidden layer* masukan pengguna. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview*.

5.2.5 Implementasi Antarmuka Solusi

Implementasi antarmuka solusi merupakan antarmuka kelima pada sistem ini. Antarmuka ini berfungsi untuk menghitung proses algoritma *extreme learning machine* terdiri dari proses *training* dan proses *testing*. Antarmuka ini juga berfungsi untuk menampilkan nilai akurasi dari masing – masing *fold*. Perhitungan yang ada didalamnya dilakukan secara komputasi, sehingga hasil yang diperoleh akan ditampilkan pada *datagridview* dalam bentuk tabel berupa hasil prediksi dan nilai akurasi. Implementasi antarmuka solusi ditunjukkan pada Gambar 5.5 sebagai berikut:



Output	Klas 1	Klas 2	Klas 3	Nilai Minimal	Prediksi	Klas
1.115713178247 98	0.115713178247983	0.884286821752017	1.88428682175202	0.115713178247983	1	1
1.26733731662586	0.267337316625863	0.732662683374137	1.73266268337414	0.267337316625863	1	1
1.40240174432876	0.402401744328756	0.597598255671244	1.59759825567124	0.402401744328756	1	1
1.40027855867884	0.400278558678837	0.599721441321163	1.59972144132116	0.400278558678837	1	1
1.23899237741676	0.238992377416757	0.761007622583243	1.76100762258324	0.238992377416757	1	1
1.17819104554181	0.178191045541807	0.821808954458193	1.82180895445819	0.178191045541807	1	1
1.30197324493423	0.301973244934234	0.698026755065766	1.69802675506577	0.301973244934234	1	1
0.910489877944492	0.0895101220555081	1.08951012205551	2.08951012205551	0.0895101220555081	1	1
1.13379803751268	0.133798037512677	0.866201962487323	1.86620196248732	0.133798037512677	1	1
1.45784930655375	0.457849306553748	0.542150693446252	1.54215069344625	0.457849306553748	1	1
1.69732093120124	0.697320931201244	0.302679068798756	1.30267906879876	0.302679068798756	2	2
1.11205549760581	0.112055497605812	0.887944502394188	1.88794450239419	0.112055497605812	1	2
1.91117866585671	0.911178665856714	0.0888213341432857	1.08882133414329	0.0888213341432857	2	2
2.15875583301395	1.15875583301395	0.158755833013948	0.841244166986052	0.158755833013948	2	2
1.76645645080411	0.766456450804105	0.233543549195895	1.23354354919589	0.233543549195895	2	2
2.57931491144437	1.57931491144437	0.579314911444369	0.420685088555631	0.420685088555631	3	2
2.01245618467887	1.01245618467887	0.012456184678868 9	0.987543815321131	0.012456184678868 9	2	2
2.1061346840071	1.1061346840071	0.1061346840071	0.8938653159929	0.1061346840071	2	2
2.06988217820265	1.06988217820265	0.0698821782026511	0.930117821797349	0.0698821782026511	2	2
2.0864476620345	1.0864476620345	0.0864476620345025	0.913552337965498	0.0864476620345025	2	2
2.58210131682791	1.58210131682791	0.582101316827908	0.417898683172092	0.417898683172092	3	3
2.17481738984373	1.17481738984373	0.174817389843728	0.825182610156272	0.174817389843728	2	3
2.34625576572735	1.34625576572735	0.346255765727352	0.653744234272648	0.346255765727352	2	3

Gambar 5.5 Implementasi Antarmuka Solusi

Gambar 5.5 menjelaskan hasil implementasi antarmuka solusi. Proses perhitungan algoritma *extreme learning machine* dilakukan pada antarmuka ini. Pengguna menekan tombol solusi, maka semua perhitungan dari awal inputan sampai akhir dilakukan. Proses perhitungan solusi melibatkan proses *load* data, normalisasi, pembagian data *training* dan data *testing*, menghitung bobot awal, dan bias. Hasilnya berupa nilai akurasi pada setiap *fold* secara bergantian sebagai data *testing*. Banyaknya *fold* yang ditampilkan berdasarkan banyaknya nilai k yang diunputkan oleh pengguna di pada antarmuka k-*fold cross validation* sebelumnya. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview*.

BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini menjelaskan hasil pengujian dan pembahasan dari penentuan kualitas air sungai menggunakan *extreme learning machine*. Pengujianyang dilakukan meliputi: pengujian nilai *k*, dan pengujian jumlah *hidden layer*.

6.1 Pengujian Nilai *k*

Metode *k-fold cross validation* merupakan metode pembagian *dataset* kedalam bentuk *fold*. Metode ini menggunakan pembagian berdasarkan nilai *k* yang merepresentasikan banyaknya *fold* yang akan dibentuk. Selain itu, nilai *k* menentukan persentase pengambilan data pada setiap *fold* menggunakan Persamaan 10. Proses pembagian data harus memenuhi ketentuan, yaitu 1) Data pada setiap *fold* tidak boleh sama, artinya setiap pengambilan data yang dilakukan pada *fold* tidak boleh dilakukan pengambilan pada *fold* lainnya. 2) Sisa bagi data diletakkan pada *fold* terakhir, artinya pengambilan data yang dilakukan pada setiap *fold* belum pasti terbagi secara rata, hal ini akan ada sisa bagi dan data tersebut diletakkan pada *fold* terakhir.

Pada pengujian nilai *k*, pengambilan data untuk setiap *fold* dilakukan secara *random*. Hal ini menimbulkan nilai akurasi yang berubah – ubah setiap dilakukan pengambilan. Maka dari itu, untuk memperoleh akurasi yang baik dilakukan percobaan pengambilan data sebanyak 10 kali untuk setiap nilai *k* dan diambil nilai terbaik dengan akurasi yang paling tinggi. Pengujian nilai k dilakukan antara 2 sampai 5 dengan jumlah data sebanyak 150. Data terdiri dari 3 kelas dengan perbandingan data yaitu 5:5:5. Pada Tabel 6.1 sampai 6.5 dijelaskan hasil pengujian nilai k sebagai berikut:

Tabel 6.1 Hasil Pengujian Nilai *k* = 2

Percobaan ke- *i*	Akurasi Data *Testing*		Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	
1	64	73	68.5
2	69.3	65.3	67.3
3	72	73.3	75.65
4	73.3	66.67	71.99
5	65.3	76	70.65
6	77.3	72	74.65
7	76	70.67	73.34
8	77.3	69.3	73.3
9	70.67	69.3	69.99
10	69.3	80	74.65
Akurasi Terbaik	**77.3**	**72**	**74.65**

Tabel 6.2 Hasil Pengujian Nilai *k* = 3

Percobaan ke- *i*	Akurasi Data *Testing*			Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	
1	83.3	62.5	77.78	52.34
2	79.17	70.83	72.2	74.1
3	75	75	72.2	74.1
4	81.25	72.92	58.52	70.9
5	72.92	77.08	74.07	74.69
6	79.17	70.83	70.37	73.46
7	68.75	77.08	77.78	74.54
8	79.17	72.97	75.93	76.02
9	72.92	66.67	81.48	73.69
10	72.92	70.83	72.2	71.74
Akurasi Terbaik	**79.17**	**72.97**	**75.93**	**76.02**

Tabel 6.3 Hasil Pengujian Nilai *k* = 4

Percobaan ke- *i*	Akurasi Data *Testing*				Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	
1	71.43	80.56	66.67	71.43	72.5
2	76.19	69.4	86.1	76.19	76.97
3	76.19	75	77.78	76.19	76.29
4	71.43	63.89	88.89	71.43	73.91
5	76.19	72.2	75	76.19	74.9
6	83.3	80.56	77.78	83.3	81.24
7	73.81	66.67	83.3	73.81	74.4
8	78.57	80.56	66.67	78.57	76.09
9	83.3	75	69.4	83.3	77.75
10	76.19	83.3	72.2	76.19	76.97
Akurasi Terbaik	**83.3**	**80.56**	**77.78**	**83.3**	**81.24**

Tabel 6.4 Hasil Pengujian Nilai *k* = 5

Percobaan ke- *i*	Akurasi Data *Testing*					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	80	70	76.67	76.67	80	76.7
2	90	60	76.67	76.67	90	78.7
3	86.67	73.3	76.67	83.3	86.67	81.32
4	80	83.3	80	76.67	80	79.99
5	90	83.3	70	83.3	90	83.32

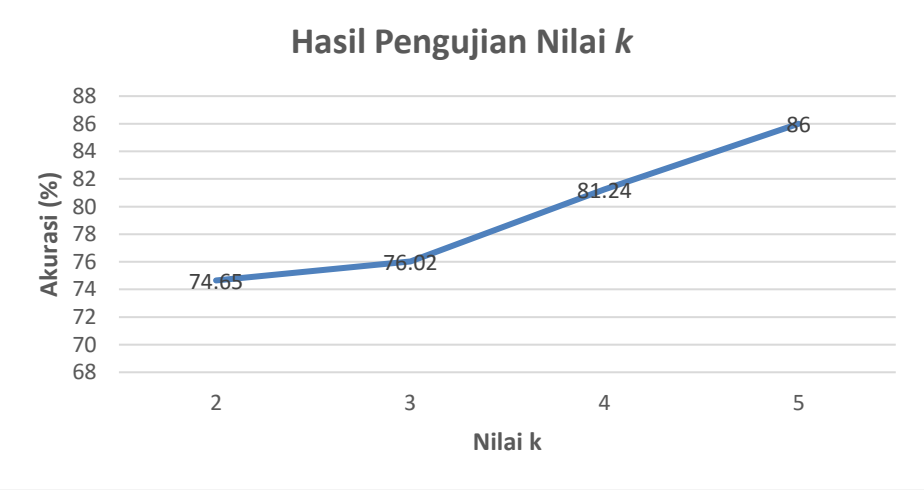
6	90	73.3	76.67	90	90	83.99
7	90	70	90	83.3	90	84.66
8	80	83.3	83.3	76.67	80	81.31
9	90	66.67	76.67	86.67	90	82
10	93.3	93.3	70	80	93.3	86
Akurasi Terbaik	**93.3**	**93.3**	**70**	**80**	**93.3**	**86**

Berdasarkan hasil pengujian nilai *k* diatas, maka diperoleh rata-rata nilai kterbaik yang ditunjukkan pada Tabel 6.5 sebagai berikut:

Tabel 6.5 Hasil Pengujian Nilai *k*

Nilai k	Data *Testing* (%)					Rata – Rata (%)
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
2	77.3	72	-	-	-	74.65
3	79.17	72.97	75.93	-	-	76.02
4	83.3	80.56	77.78	83.3	-	81.24
5	93.3	93.3	70	80	93.3	86

Grafik hasil pengujian nilai *k* pada metode *k-fold cross validation* ditunjukkan pada Gambar 6.1 sebagai berikut:



Gambar 6.1 Grafik Hasil Pengujian Nilai *k*

Nilai *k* pada metode k-*fold cross validation* berfungsi dalam membentuk banyaknya *fold* sebagai kelompok data. Nilai *k* juga digunakan dalam menentukan persentase pengambilan data pada setiap *fold* menggunakan Persamaan 11. Pengelompokkan data dalam bentuk *fold* digunakan dalam menentukan data *training* dan data *testing* dimana salah satunya diantara *fold* yang lain harus pernah menjadi data *testing*. Nilai *k* semakin tinggi, semakin

banyak pula *fold* yang dibentuk dan semakin kecil pula persentase pengambilan datanya. Semakin besar nilai *k*, variasi dari estimasi data dalam setiap *fold* yang dihasilkan berkurang karena nilai *k* semakin besar. Apabila data pada *fold* sedikit, maka setiap *fold* yang digunakan sebagai data *testing* secara bergantian memiliki peluang menghasilkan nilai akurasi yang baik. Hal ini dikarenakan banyaknya data *testing* semakin sedikit dibandingkan data *training*. Selain itu, nilai *k* juga menentukan banyaknya iterasi dalam percobaan yang nantinya diperoleh rata – rata akurasi sehingga nilai *k* yang besar memiliki banyak percobaan dengan akurasi disetiap percobaannya.

Berdasarkan grafik hasil pengujian nilai *k* pada Gambar 6.1 diperoleh nilai akurasi terbaik 86% pada nilai *k* = 5 (Lihat Lampiran 5). Nilai *k* yang besar akan membentuk banyak *fold* dan persentase pengambilan data pada setiap *fold* semakin kecil sehingga data yang ada pada *fold* semakin sedikit jumlahnya. Kondisi ini memungkinkan data *testing* yang digunakan semakin sedikit jumlahnya dibandingkan data *training*. Hal ini akan meningkatkan nilai akurasi karena perbandingan data *training* lebih besar daripada data *testing*. Pada grafik pengujian nilai *k* dijelaskan bahwa semakin besar nilai *k* maka semakin tinggi akurasi yang diperoleh. Sehingga dapat disimpulkan bahwa dengan nilai *k* yang besar, pada penelitian ini menghasilkan data *testing* yang semakin sedikit jumlahnya agar hasil akurasi semakin tinggi.

6.2 Pengujian Jumlah *Hidden Node*

Pada metode *extreme learning machine* bagian *input layer* terhubung dengan *hidden layer*. *Hidden layer* merupakan bagian yang memproses perhitungan dari *input* menjadi *output*. Secara struktur bagian *hidden layer* terdiri dari beberapa *node* yang menghubungkan *input layer* dengan *output layer* berada ditengah – tengah diantara keduanya. Penghubung (*connector*) yang menghubungkan dengan *input layer* disebut bobot *input* (*w*) dan penghubung dengan *output layer* disebut bobot *output* (β). Setiap penghubung memiliki tingkat kekuatan (*value*) yang berbeda. Bobot *input* digunakan dalam proses *training* yang dilakukan pada bagian *hidden layer*. *Hidden layer* juga berfungsi sebagai pembentuk bobot awal dan bias. Banyaknya *node* pada *hidden layer* mempengaruhi proses perhitungan dari *input layer* yang terdiri dari *input node* menghasilkan keluaran dari *hidden node*. *Output* dari *hidden node* akan diteruskan pada *output layer* yang dihubungkan dengan bobot *output*. Bobot *output* sendiri merupakan hasil keluaran dari *hidden layer* supaya bisa terhubung dengan *output layer*.

Pada proses *training*, jumlah *hidden node* digunakan untuk pembentukan bobot *input* dan bias. Pembentukan bobot *input* diperoleh dari Persamaan 1 yang disusun oleh banyaknya *hidden node* dan *input layer*, sedangkan pembentukan bias diperoleh dari Persamaan 2 yang disusun oleh *hidden node*. Interval bobot *input* dan bias yaitu rentang nilai 0 sampai dengan 1 dan banyaknya *hidden node* pada pengujian ini antara 2 sampai dengan 7. Pada pengujian jumlah *hidden layer*data *training* dan data *testing* yang digunakan

diambil dari hasil pengujian nilai k terbaik pada pengujian sebelumnya, yaitu 5. Pengujian jumlah *hidden layer* digunakan dalam pembentukan bobot *input* dan bias. Pembentukan bobot *input* dan bias dilakukan sebanyak 10 kali percobaan secara acak untuk setiap nilai *hidden layer* dan diambil nilai akurasi terbaik. Pada Tabel 6.6 sampai 6.12 dijelaskan hasil pengujian jumlah *hidden layer* sebagai berikut:

Tabel 6.6 Hasil Pengujian *Hidden Node* = 2

Percobaan ke- *i*	Akurasi Nilai *k*=5					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	73.3	66.7	63.3	76.67	73.3	70.65
2	73.3	73.3	66.67	70	73.3	71.31
3	80	66.67	53.3	80	80	71.99
4	70	56.67	73.3	73.3	70	68.65
5	73.3	63.3	66.67	73.3	73.3	69.97
6	66.67	60	86.67	66.67	66.67	69.34
7	66.67	73.3	63.3	66.67	66.67	67.32
8	70	76.67	50	66.67	70	66.67
9	80	60	63.3	80	80	72.6
10	76.67	70	66.67	76.67	76.67	73.34
Akurasi Terbaik	**76.67**	**70**	**66.67**	**76.67**	**76.67**	**73.34**

Tabel 6.7 Hasil Pengujian *Hidden Node* = 3

Percobaan ke- *i*	Akurasi Nilai *k*=5					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	73.3	60	76.67	80	73.3	72.65
2	83.3	70	60	76.67	83.3	74.65
3	73.3	73.3	80	73.3	73.3	74.64
4	73.3	83.3	73.3	73.3	73.3	75.3
5	76.67	73.3	76.67	76.67	76.67	76
6	76.67	83.3	73.3	76.67	76.67	77.32
7	83.3	63.3	76.67	83.3	83.3	77.97
8	83.3	66.67	80	80	83.3	78.5
9	86.67	73.3	76.67	86.67	86.67	82
10	90	76.67	70	86.67	90	82.67
Akurasi Terbaik	**90**	**76.67**	**70**	**86.67**	**90**	**82.67**

Tabel 6.8 Hasil Pengujian *Hidden Node* = 4

Percobaan ke- *i*	Akurasi Nilai *k*=5					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	80	63.3	83.3	73.3	80	76
2	80	66.67	76.67	86.67	80	78
3	80	76.67	66.67	80	80	76.67
4	80	73.3	76.67	73.3	80	76.65
5	80	76.67	80	80	80	79.33
6	83.3	76.67	76.67	76.67	83.3	79.32
7	80	70	70	80	80	76
8	83.3	76.67	76.67	70	83.3	78
9	86.67	73.3	83.3	86.67	86.67	83.32
10	86.67	76.67	76.67	86.67	86.67	82.67
Akurasi Terbaik	**86.67**	**73.3**	**83.3**	**86.67**	**86.67**	**83.32**

Tabel 6.9 Hasil Pengujian *Hidden Node* = 5

Percobaan ke- *i*	Akurasi Nilai *k*=5					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	80	73.3	73.3	86.67	80	78.65
2	83.3	80	83.3	76.67	83.3	81.31
3	86.67	80	73.3	76.67	86.67	80.66
4	80	86.67	76.67	73.3	80	79.33
5	80	86.67	76.67	73.3	80	79.33
6	80	73.3	83.3	80	80	79.32
7	83.3	76.67	80	83.3	83.3	81.25
8	90	70	80	83.3	90	82.66
9	93.3	76.67	70	86.67	93.3	84
10	93.3	73.3	73.3	93.3	93.3	85.3
Akurasi Terbaik	**93.3**	**73.3**	**73.3**	**93.3**	**93.3**	**85.3**

Tabel 6.10 Hasil Pengujian *Hidden Node* = 6

Percobaan ke- *i*	Akurasi Nilai *k*=5					Rata – Rata Akurasi (%)
	Fold 1 (%)	Fold 2 (%)	Fold 3 (%)	Fold 4 (%)	Fold 5 (%)	
1	83.3	73.3	80	83.3	80.3	80.64

2	86.67	66.67	76.67	86.67	86.67	80.87
3	86.67	80	76.67	83.3	86.67	82.66
4	80	96.67	66.67	80	80	80.67
5	80	83.3	76.67	80	80	79.99
6	86.67	83.3	66.67	86.67	86.67	82
7	90	83.3	76.67	93.3	90	86.65
8	86.67	66.67	76.67	83.3	86.67	80
9	83.3	70	76.67	86.67	83.3	80
10	93.3	66.67	76.67	90	93.3	84
Akurasi Terbaik	**90**	**83.3**	**76.67**	**93.3**	**90**	**86.65**

Tabel 6.11 Hasil Pengujian *Hidden Node* = 7

Percobaan ke- *i*	**Akurasi Nilai *k*=5**					**Rata – Rata Akurasi (%)**
	Fold 1 (%)	**Fold 2 (%)**	**Fold 3 (%)**	**Fold 4 (%)**	**Fold 5 (%)**	
1	86.67	66.67	83.3	86.67	86.67	82
2	83.3	73.3	73.3	83.3	83.3	79.3
3	83.3	86.67	70	83.3	83.3	81.31
4	90	73.3	80	86.67	90	83.99
5	83.3	76.67	93.3	86.67	83.3	83.97
6	86.67	86.67	70	90	86.67	84
7	83.3	80	86.67	83.3	83.3	82.64
8	83.3	80	86.67	83.3	83.3	83.31
9	90	83.3	86.67	70	90	83.99
10	93.3	76.67	83.3	93.3	93.3	87.97
Akurasi Terbaik	**93.3**	**76.67**	**83.3**	**93.3**	**93.3**	**87.97**

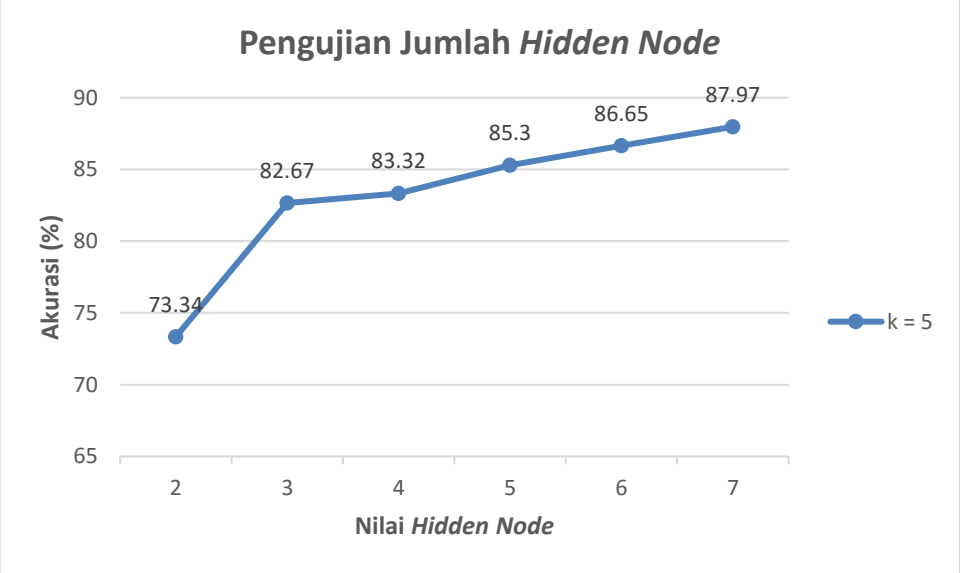
Berdasarkan hasil pengujian jumlah *hidden node* diatas, maka diperoleh rata-rata nilai *hidden node* terbaik yang ditunjukkan pada Tabel 6.12 sebagai berikut:

Tabel 6.12 Hasil Pengujian Jumlah *Hidden Node*

Jumlah *Hidden Node*	**Nilai *k* = 5**					**Rata – Rata (%)**
	Fold 1 (%)	**Fold 2 (%)**	**Fold 3 (%)**	**Fold 4 (%)**	**Fold 5 (%)**	
2	76.67	70	66.67	76.67	76.67	73.34
3	90	76.67	70	86.67	90	82.67
4	86.67	73.3	83.3	86.67	86.67	83.32

5	93.3	73.3	73.3	93.3	93.3	85.3
6	90	83.3	76.67	93.3	90	86.65
7	93.3	76.67	83.3	93.3	93.3	87.97

Grafik hasil pengujian jumlah *hidden node* ditunjukkan pada Gambar 6.2 sebagai berikut:



Gambar 6.2 Grafik Hasil Pengujian Jumlah *Hidden Node*

Hidden node pada metode *extreme learning machine* terdiri dari *node-node* merupakan unit yang melakukan proses perhitungan yang mengolah *input* menjadi *output*. Hal ini dikarenakan, *hidden node* bagian terpenting yang mengolah semua masukan yang nantinya menjadi keluaran. Selain itu, *hidden layer* memiliki parameter yang menghubungkan antarlayer (Shamshirband dkk, 2009). Parameter – parameter tersebut dibentuk berdasarkan banyaknya *hidden node*. Setiap penghubung memiliki tingkat kekuatan yang berbeda (nilai yang berbeda) (Huang dkk, 2005). Hal ini memungkinkan adanya perbedaan hasil yang dicapai oleh setiap unit *hidden node*.

Berdasarkan grafik hasil pengujian jumlah *hidden node* pada Gambar 6.2 diperoleh nilai akurasi terbaik 87.97% pada jumlah *hidden node* sebanyak 7 (Lihat Lampiran 5). Semakin besar jumlah *hidden node*-nya, maka nilai akurasi yang diperoleh semakin tinggi. Nilai *hidden node* yang besar akan membentuk banyak penghubung (*connector*) dengan *input layer* dan*output layer*. Kondisi ini memungkinkan unit pemroses pada sistem yang melakukan proses pembobotan untuk mengenali data memiliki kemampuan yang baik dengan semakin banyaknya pertimbangan keputusan yang dilakukan oleh *hidden node*.Pada grafik pengujian jumlah *hidden layer* dijelaskan bahwa semakin banyak *hidden node*-nya maka, semakin tinggi akurasi yang diperoleh. Sehingga dapat disimpulkan bahwa dengan jumlah *hidden layer* yang besar, pada penelitian ini

seleksi hasil terhadap keluaran *hidden layer* yang masuk pada *output layer*semakin detail, sehingga tingkat ketepatan hasil akan semakin tinggi.

6.3 Pembahasan Hasil Pengujian

Penentuan kualitas air sungai menggunakan metode *extreme learning machine* dilakukan dengan menggunakan parameter dengan hasil yang optimum yang diperoleh melalui hasil skenario pengujian yaitu, nilai *k* = 5 dan jumlah *hidden layer* = 7. Dengan menggunakan nilai dari parameter tersebut, sistem memperoleh solusi penyelesaian nilai akurasi yang optimum sebesar 87.97%. Kondisi ini belum sesuai dengan hasil yang diperoleh menggunakan metode standar baku dari pemerintah. Hal ini dikarenakan oleh beberapa alasan. Pertama, metode STORET merupakan metode standar dari pemerintah yang melakukan perhitungan secara manual dengan pemberian skor dengan rentang nilai tertentu pada setiap nilai parameter air sesuai kebijakan pemerintah yang kurang representatif dengan kondisi dilapangan. Kedua, sistem menggunakan 7 parameter utama kimia-fisika yang paling berpengaruh, sedangkan pada metode STORET menggunakan beberapa parameter tambahan, sehingga hasil penentuannya kurang obyektif dengan metode STORET, hal ini dikarenakan keterbatasan alat, waktu, dan biaya dalam hasil analisa komponen di laboratorium. Ketiga, perhitungan menggunakan metode STORET pada nilai baku parameter pengukurnya kurang sesuai dengan kondisi di lapangan, hal ini air sungai tidak hanya dalam satu tempat saja melainkan mengalir dan melewati daerah yang tingkat pencemarannya pun berbeda – beda.

Terlepas dari paparan beberapa alasan diatas, sistem yang dibangun bertujuan sebatas memberikan rekomendasi penentuan kualitas air sungai berdasarkan pendekatan *extreme learning machine* yang dievaluasi dengan parameter yang menghasilkan nilai akurasi paling baik. Selanjutnya, dari hasil pnentuan tersebut dapat dilakukan analisis lebih lanjut guna menemukan informasi tertentu sehingga dapat digunakan sebagai bahan pertimbangan dalam menentukan kualitas air sungai yang sesuai dengan kondisi air yang sebenarnya.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan pembahasan dari penentuan kualitas air sungai menggunakan metode *extreme learning machine* maka diperoleh kesimpulan sebagai berikut:

1. Algoritma *extreme learning machine* dapat digunakan dalam penentuan kualitas air sungai. Hal tersebut dilakukan dengan mendefinisikan jumlah *hidden layer* sebagai *layer* yang melakukan perhitungan menghasilkan keluaran dan melakukan pembagian data *training* dan data *testing* yang ideal menggunakan metode *k-fold cross validation*. Pada *hidden layer* terdiri dari *node-node* yang disebut *hidden node* berfungsi sebagai unit pemroses yang saling terhubung dengan kedua *layer* lainnya. Pada *input layer*, data masukan akan terhubung dengan semua *hidden node* melalui parameter penghubung pada setiap *input node*. Data akan diproses menjadi keluaran yang akan diteruskan ke *layer* berikutnya. Hasil keluaran dari setiap *hidden node* berupa fungsi keluaran *hidden layer* dan parameter penghubung (β) yang akan diteruskan ke *output layer*, dimana semua *hidden node* terhubung pada setiap *output node*. Pada tahap akhir dari *output layer* akan dilakukan evaluasi penentuan yang akan dihitung akurasi dengan acuan hasil diagnosa pakar menggunakan STORET.
2. Penambahan jumlah *hidden node* berpengaruh terhadap akurasi hasil penentuan kualitas air. Selain itu, dalam proses pembagian data *training* dan data *testing* menggunakan *k-fold cross validation* ditentukan berdasarkan nilai *k*. Hal ini akan berpengaruh terhadap banyaknya data yang digunakan untuk proses *training* dan *testing*. Berdasarkan hasil pengujian, nilai akurasi tertinggi diperoleh dengan nilai *k* = 5 dan jumlah *hidden node* = 7 mencapai 87.89%.

7.2 Saran

Penelitian tentang penentuan kualitas air sungai menggunakan metode *extreme learning machine* dapat dikembangkan dengan beberapa saran sebagai berikut:

1. Pada penelitian mendatang dilakukan penambahan seleksi fitur bertujuan untuk memilih fitur yang berpengaruh dari keseluruhan parameter air sungai, sehingga kondisi dimana fitur yang berpengaruh selain fitur utama (TSS, BOD, COD, DO, pH, Fenol, Minyak dan Lemak) digunakan dalam proses perhitungan.
2. Pada penelitian mendatang menggabungkan algoritma/metode lain dengan ELM seperti metode optimasi atau dengan metode kecerdasan buatan lainnya supaya didapatkan bobot awal yang optimal. Hal ini bertujuan agar hasil yang diperoleh lebih baik daripada penelitian sebelumnya.

DAFTAR PUSTAKA

Agustiningsih, Dyah. 2014. Thesis: Analisis Kualitas Air Dan Strategi Pengendalian Pencemaran AirSungai Blukar Kabupaten Kendal. Vol. 9 No.2 September 2012, ISSN 1907-187X.

Anguita, Davide., Ghio, Alessandro., Ridella, Sandro., Sterpi Dario. 2009. K–*Fold Cross Validation for Error Rate Estimate in Support Vector Machines*. University of Genova, Italy.

Cao, Jiuwen dan Xiong, Lianglin. 2014. Protein Sequence Classification with Improved Extreme Learning Machine Algorithms. China.

Fardani, Delia Putri. 2015. Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode Extreme Learning Machine (Studi Kasus : Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto). Vol. 1, No. 1, April 2015. Universitas Airlangga.

Juantari, Gilang Y. 2013. Status Trofik Dan Daya Tampung Beban Pencemaran Waduk Sutami. Vol 4. No 1 pp 61-66.

Hermawan, A. 2006. Jaringan Saraf Tiruan Teori dan Aplikasi. Andi: Yogyakarta.

Huang, G.B., Zhu, Q.Y., dan Siew, C.K. 2004.Extreme Learning Machine : *A New LearningScheme of Feddforward neural Networks.Proceeding of International Joint Conferenceon Neural Networks*. Hungary, 25-29 Juli.

Huang, G.B., Zhu, Q.Y., dan Siew, C.K. 2005.*Extreme Learning Machine : Theory andapplications. Elsevier science : Neurocomputing*70 (2006) 489-501.

Keputusan Menteri Negara Lingkungan Hidup Nomor 115 Tahun 2003 tentang Pedoman Penentuan Status Mutu Air. Menteri Negara Lingkungan Hidup. Jakarta.

Paramadyastha, Annisah. 2011. Studi Penentuan Status Mutu Air dengan Metode Storet dan Metode Indek Pencemaran di Waduk Sutami. Malang. Teknik Pengairan Universitas Brawijaya.

Peraturan Pemerintah Nomor 38 Tahun 2011 tentang Sungai.

Prastyo, DedyDwi., 2010. Peramalan Menggunakan Metode Eksponensial Smoothing, Jurusan Statistika Institut Teknologi Sepuluh November, Surabaya.

Santosh, Kumar Sahu, Sauravranjan, Sarangi and Sanjaya, Kumar Jena, "A Detail Analysis on Intrusion DetectionDatasets," in International Advance Computing Conference (IACC), 2014.

Shamshirband, Shahaboddin., Mohammadi, Karsa.,Tog, Chong Wen., Petkovic, Dalibor., Porcu, Emilio., Mostafaeipour, Ali., Ch, Sudheer., dan Sedaghat, Ahmat. 2015. *Application of extreme learning machine for estimation of wind speed distribution*. Springer-Verlag Berlin Heidelberg.

Siang, J. (2005). Jaringan Saraf Tiruan & Pemrogramannya Menggunakan Matlab.Yogyakarta: Andi.

Sun, Z.L., Choi, T.M., Au, K.F., dan Yu, Y. 2008.*Sales Forecasting using Extreme Learning Machine with Application in Fashion Retailing.Elsevier Decision Support Systems* 46 (2008) 411-419.

Suriawiria, Unus. 2003. Air dalam Kehidupan dan Lingkungan yang Sehat. Penerbit Alumni. Bandung

Tafangenyasha, C. and T. Dzinomwa. 2005. *Land-use Impacts on River Water Quality in Lowveld Sand River Systems in South-East Zimbabwe. Land Use and Water Resources Research* 5 : 3.1-3.10. http://www.luwrr.com

Wiwoho, 2005, Model Identifikasi Daya Tampung Beban Cemaran Sungai Dengan QUAL2E. Tesis. Universitas Diponegoro. Semarang

DAFTAR LAMPIRAN

Lampiran 1 Data Kualitas Air Sungai Menggunakan Metode STORET

No	FISIKA	KIMIA Anorganik				KIMIAOrganik		Metode STORET	
	TSS	BOD	COD	DO	pH	Fenol	Minyak & Lemak	Skor	Keterangan (Tercemar)
1.	0	15.35	56.8	11.2	7.5	0.0775	0.8	-30	Sedang
2.	0	10	34.8	15.3	8.05	0.0145	4.5	-42	Berat
3.	44.85	12.6	35.4	18.55	8.85	0.1185	4.25	-39	Berat
4.	72.75	19.75	62.85	10.45	8.1	0.172	0	-32	Berat
5.	10	6.2	13.9	10.15	7.55	0.172	1.5	-30	Sedang
6.	6.2	11.3	35.1	16.925	8.45	0.086	1.425	-30	Sedang
7.	24.25	6.95	15.6	10.85	7.3	0.067	0.8	-20	Sedang
8.	26.25	7.9	21	11.75	7	0.196	0	-22	Sedang
9.	17.4	6.9	16.3	7.95	6.8	0.128	0.9	-20	Sedang
10.	21.25	6.25	13.4	7.45	6.95	0.0925	0.8	-20	Sedang
11.	17.75	5.05	20.6	3.5	6.9	0.0885	0.8	-30	Sedang
12.	25.35	4.1	16.85	6.75	7	0.151	3.5	-30	Sedang
13.	35	2.5	11.2	8.9	7.4	0.188	0.8	-6	Ringan
14.	229.8	3.85	20.7	7.2	6.95	0.1135	0	-22	Sedang
15.	41.3	12.8	41.45	9.7	7.95	0.0915	0	-30	Sedang
16.	69.6	21.3	97.15	13.2	8.5	0.119	0	-31	Berat
17.	23.7	12.15	32.5	11.95	8.2	0.008	0	-28	Sedang
18.	17.8	3.4	10.25	6.8	7.45	0.176	0	-20	Sedang
19.	7.1	3.6	9.85	5.85	7.35	0.104	1.3	-26	Sedang
20.	15.7	5.85	19.6	6.7	7.2	0.187	2.9	-26	Sedang
21.	15.55	3.25	9.25	7.6	6.95	0.1705	0.4	-18	Sedang
22.	25.55	2.85	14.15	8.05	7	0.017	0.4	-16	Sedang
23.	54.9	7.7	47	8.6	7.35	0.104	3.125	-54	Berat
24.	17.75	6.35	10.5	7.9	7.45	0.0885	0	-30	Sedang
25.	32.1	4.8	20.85	8.35	7.8	0.151	0.5	-20	Sedang
26.	27.6	7.15	20.85	10.2	7.65	0.0975	0.8	-22	Sedang
27.	54.3	14.4	27.3	11.2	7.55	0.1	3	-38	Berat
28.	177.65	2.4	12.05	4.25	6.55	0.1575	1.5	-21	Sedang
29.	18.1	3.65	9.9	6.9	7.3	0.3825	6.85	-28	Sedang
30.	8	3.95	9.05	8.9	7.6	0.334	0	-20	Sedang
31.	11.9	4.5	9.8	7.7	7.5	0.126	4.25	-22	Sedang
32.	13.65	4.8	9.3	6.3	7.3	0.046	0	-16	Sedang
33.	26.9	5.15	11.75	6.2	7.55	0.095	3.5	-22	Sedang
34.	8.3	5.7	17.7	6	7.95	0.126	0	-16	Sedang
35.	25.8	6.25	23.35	12.7	8.2	0.0685	2.85	-32	Berat

36.	13.15	6.35	21.25	7.6	7.4	0.029	2.25	-24	Sedang
37.	66.9	5.3	10.675	5.6	7	0.087	4.1	-35	Berat
38.	87.35	3.7	25.8	7.4	8.1	0.1225	2.9	-37	Berat
39.	84.5	2.5	8.25	10.3	8.25	0.058	5	-24	Sedang
40.	33.6	2.5	8.75	6.15	7.55	0.039	0.39	-10	Ringan
41.	72	3.8	10.85	5.45	7.45	0.029	6.3	-31	Berat
42.	43.65	7.95	53	12.1	9	0.081	0	-29	Sedang
43.	14	4.85	10.3	5.65	7.6	0.127	6	-26	Sedang
44.	18.95	2.45	8.8	6	7.45	0.062	3.5	-20	Sedang
45.	16.5	2.75	11.95	5.8	7.45	0.0735	3.6	-22	Sedang
46.	45.05	4.25	26.1	7.35	7.8	0.212	2.85	-35	Berat
47.	28.1	5.4	17.5	8.25	8.25	0.079	3.25	-28	Sedang
48.	7.05	4.2	13.1	6.4	7.6	0.036	3	-22	Sedang
49.	22.9	4.4	16.15	9.15	7.95	0.007	2.7	-22	Sedang
50.	203.9	4.7	13.205	9.1	7.2	0.008	4.25	-31	Berat
51.	27.3	5.8	27.803	7.55	8.05	0	4	-28	Sedang
52.	48.35	4.5	33.0675	7.7	7.9	0	3.5	-27	Sedang
53.	30.85	5	16.2775	10.4	8.75	0.0027	4.55	-26	Sedang
54.	13	3.325	11.652	8.9	8.15	0	3.95	-22	Sedang
55.	20.15	1.925	6.408	7.85	7.85	0	2.85	-10	Ringan
56.	38.5	3.075	12.9275	6.3	7.1	0	1.9	-19	Sedang
57.	17.25	7.275	27.0195	8.8	6.9	0	3.75	-28	Sedang
58.	42.3	3.75	9.7375	42.25	7.1	0	4.75	-31	Berat
59.	25.55	3.375	12.678	5.5	7.25	0	5	-20	Sedang
60.	35.1	3.55	18.864	7	7.15	0	2.5	-16	Sedang
61.	13.3	8.2	15.6	7.2	7	0.0001	0.3	-12	Sedang
62.	20.7	14.6	21.2	6.5	7.1	0.0001	1	-14	Sedang
63.	21	8.9	18.8	4.9	7	0.0001	0.83	-14	Sedang
64.	16.3	9.8	20.2	4.9	7.1	0.0019	0.83	-22	Sedang
65.	25	13.8	25.2	6	7.4	0.0001	1	-22	Sedang
66.	17	14.4	22.3	4.2	6.8	0.1094	1.25	-30	Sedang
67.	24.3	4.3	8.3	9.3	6.6	0.0001	1	-10	Ringan
68.	22.7	3	7.3	13.7	6.9	0.0001	0	-4	Ringan
69.	19	3	56.8	7.95	7.35	0	0.8	-8	Ringan
70.	17	2.5	34.8	7.45	7.2	0	0	-10	Ringan
71.	20	2.85	35.4	13	6.95	0	0.9	-6	Ringan
72.	44.85	2.4	62.85	6.3	7	0	0.8	-2	Ringan
73.	10	2.5	41.45	6.2	7.35	0	0.8	-10	Ringan
74.	6.2	2.5	97.15	6	7.45	0	0.4	-10	Ringan
75.	24.25	1.925	27.803	12.7	7.8	0.0001	0.4	-10	Ringan
76.	17.4	8.2	17.5	7.6	7.65	0.0001	0	-10	Ringan
77.	21.25	14.6	13.1	5.6	7.55	0.0001	0.5	-10	Ringan
78.	17.75	2.9	16.15	7.4	6.55	0.039	0.8	-8	Ringan

79.	25.35	1.8	13.205	10.3	7.3	0.029	0	-10	Ringan
80.	35	1.8	11	12	7.5	0.081	0.5	-6	Ringan
81.	41.3	14.4	11	18.55	8.05	0.0001	0.8	-2	Ringan
82.	18.5	1.09	25.2	10.45	8.85	0.0001	0.3	-6	Ringan
83.	28.1	2.1	25.2	10.15	8.1	0	0.83	-10	Ringan
84.	7.05	5	8	16.925	7.55	0	0.83	-10	Ringan
85.	22.9	2.45	25.8	10.85	6.55	0.0001	1	-8	Ringan
86.	23	2.75	53	10	7.3	0.0001	0	-10	Ringan
87.	30	4.25	8.25	8.9	7.6	0.0001	1	-6	Ringan
88.	22	5.4	10	7.3	7.5	0.0001	0.3	-2	Ringan
89.	23.7	4.2	7	7.95	7.3	0.0001	1	-10	Ringan
90.	17.8	2.5	16.2775	7.45	8.05	0.0001	0.83	-4	Ringan
91.	7.1	2.5	11.652	6.8	8.85	0	0.83	-2	Ringan
92.	15.7	3	6.408	5.85	8.1	0	1	-6	Ringan
93.	15.55	3	27.803	6.7	7.55	0	0.8	-10	Ringan
94.	25.55	3.75	10.85	7.6	8.45	0	0	-6	Ringan
95.	32	3.375	10.3	8.05	7.3	0.0001	0.9	-2	Ringan
96.	27	3.55	8.8	8.6	7	0.0001	0.8	-6	Ringan
97.	14	8.2	11.95	7.9	6.8	0	0.8	-10	Ringan
98.	18.95	14.6	10.5	8.35	6.95	0	0.4	-10	Ringan
99.	16.5	8.9	20.85	10.2	6.9	0.0001	0.45	-10	Ringan
100.	45.05	6.35	20.85	11.2	7	0.0001	0.84	-10	Ringan
101.	16	3.95	9.25	12.7	7.45	0	0.5	-6	Ringan
102.	18.1	4.5	15.6	7.6	7.35	0.0001	0.8	-10	Ringan
103.	8	4.1	21.3	5.6	7.2	0.0001	1	-8	Ringan
104.	11.9	15.35	16.3	7.4	8.45	0.0001	1	-10	Ringan
105.	13.65	10	13.4	12	7.3	0	0.8	-6	Ringan
106.	26.9	12.6	20.6	4.25	7	0	0	-2	Ringan
107.	8.3	19.75	16.85	6.9	6.8	0	0.5	-10	Ringan
108.	25.8	6.2	11.2	8.9	6.95	0	0.8	-10	Ringan
109.	13.15	11.3	20.7	7.7	6.9	0.0001	0.3	-10	Ringan
110.	21	6.95	11	6.3	7	0.0001	1	-10	Ringan
111.	28.1	7.9	9.5	6.2	7.4	0.0001	0	-8	Ringan
112.	7.05	2	91.56	6	6.95	0.0001	1	-10	Ringan
113.	22.9	5.15	10.25	12.7	7.95	0	0.3	-6	Ringan
114.	16	5.7	9.85	7.6	8.5	0.0001	1	-2	Ringan
115.	155	13.15	44	29	7.5	0.0775	6	-35	Berat
116.	117	8.5	47	10	8.05	0.0145	3.5	-45	Berat
117.	18	6.7	30	35	8.85	0.1185	3.6	-36	Berat
118.	1	13.5	46	35	8.1	0.172	2.85	-38	Berat
119.	178	4.5	88	38	7.55	0.172	3.25	-33	Berat
120.	115	16.25	74	32	8.45	0.086	3	-33	Berat
121.	67	10.5	55	41	7.3	0.067	2.7	-33	Berat

122.	83	6.8	84	32	7	0.196	4.25	-45	Berat
123.	181	20.15	29	40	6.8	0.128	4	-45	Berat
124.	93	7.7	85	22	6.95	0.0925	3.5	-35	Berat
125.	65	7	94	14	6.9	0.0885	4.55	-43	Berat
126.	172	14.8	81	35	7	0.151	3.95	-43	Berat
127.	186	16.4	18	40	7.4	0.188	2.85	-33	Berat
128.	192	7.76	30	22	6.95	0.1135	1.9	-43	Berat
129.	95	4.8	65	30	7.95	0.0915	3.75	-45	Berat
130.	32	129	75	35	8.5	0.119	4.75	-38	Berat
131.	148	7	62	27	8.2	0.008	5	-43	Berat
132.	94	12	90	20	7.45	0.176	2.5	-40	Berat
133.	45	6	85	21	7.35	0.104	0.3	-38	Berat
134.	64	10.43	14	22	7.8	0.151	1	-32	Berat
135.	154	15.4	94	13	7.65	0.0975	0	-35	Berat
136.	40	4.9	21	29	7.55	0.1	1.2	-32	Berat
137.	160	14.9	73	32	6.55	0.1575	0	-34	Berat
138.	0	7.2	63	19	7.3	0.3825	1.5	-36	Berat
139.	177	6	52	7	7.6	0.334	0.8	-36	Berat
140.	96	15	58	11	7.5	0.126	0.8	-38	Berat
141.	32	15.5	17	11	7.3	0.046	0.4	-33	Berat
142.	48	6.35	65	16	7.55	0.095	0.4	-33	Berat
143.	104	21.7	86	19	7.95	0.126	0	-33	Berat
144.	81	20.1	80	6	8.2	0.0685	0.5	-45	Berat
145.	130	11	23	30	7.4	0.029	0.8	-45	Berat
146.	55	19	18	19	7	0.087	0	-34	Berat
147.	54	6	85	33	8.1	0.1225	0.5	-33	Berat
148.	73	19	14	10	8.25	0.058	0.8	-33	Berat
149.	24	18	77	20	7.55	0.039	0.3	-33	Berat
150.	119	5	82	30	7.45	0.029	0.83	-35	Berat

Mengetahui,

Dosen Pakar, Teknik Pengairan

Fakultas Teknik, Universitas Brawijaya

Dr. Eng. Riyanto Haribowo, ST., MT.

NIP: 19770424 200312 1 001

Lampiran 2 Surat Permohonan Pakar

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
FAKULTAS ILMU KOMPUTER
Gedung A FILKOM Lt. 1, Jl. Veteran No 8 Malang, 65145, Indonesia
Telp : +62-341-577911, Fax : +62-341-577911
http : filkom.ub.ac.id E-mail : ptiik@ub.ac.id

Nomor : 3095 / UN10.36/AK/2016
Perihal : ***Permohonan menjadi pakar untuk kebutuhan skripsi***

Yth. Dekan Fakultas Teknik
Universitas Brawijaya

Untuk mendukung penyusunan skripsi mahasiswa berikut :

Nama	:	**Alvia Nur Azizah**
NIM	:	**125150201111010**
Judul Skripsi	:	**Penentuan Kualitas Air Sungai Menggunakan Extreme Learning Machine**
Dosen Pembimbing	:	**1. Imam Cholissodin, S.Si.,M.Kom** **2. Edy Santoso, S.Si, M.Kom**
Prodi	:	**Informatika/ Ilmu Komputer**

Mohon kesediaan salah satu dosen dari Jurusan Pengairan untuk menjadi pakar dalam penyelesaian skripsi mahasiswa tersebut

Atas perhatian dan kerjasamanya kami ucapkan terima kasih.

a.n. Dekan
Kepala Tata Usaha Fakultas Ilmu Komputer,

Dra. Ekasang Budha Mahadyawati, M.AP.
NIP. 19581213 19820 2 2001

Dosen Pembimbing I

Imam Cholissodin, S.Si.,M.Kom.
NIP 201201 850719 1 001

Tembusan Kepada Yth:
1. *Ketua Program Studi Informatika / Ilmu Komputer*
2. *Ketua Jurusan Pengairan Fakultas Teknik UB*
3. *Mahasiswa yang bersangkutan*

Lampiran 3 Surat Ketersediaan Pakar

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK PENGAIRAN
Jl. Mayjend. Haryono no. 167, Malang, 65145, Indonesia
Telp & Fax. : +62-341-562454
http://pengairen.ub.ac.idE-mail : pengairan@ub.ac.id

SURAT KETERANGAN PENUGASAN
NO : 394 /UN10.6.3/KP/2016

Ketua Jurusan Teknik Pengairan Fakultas Teknik Universitas Brawijaya, dengan ini menugaskan kepada dosen Jurusan Teknik Pengairan tersebut di bawah ini :

Nama	: Dr.Eng. Riyanto Haribowo,ST., MT.
NIP	: 19770424 200312 1 001
Pangkat/Gol	: Penata / IIIi/c
Jabatan	: KaLab Perencanaan Bangunan Air

untuk menjadi pakar dalam penyelesaian skripsi mahasiswa :

Nama	: Alvia Nur Azizah
NIM	: 125150201111010
Prodi	: Informatika / Ilmu Komputer
Judul Skripsi	: Penentuan Kualitas Air Sungai Menggunakan *Extreme Learning Machine*

Demikian surat keterangan penugasan ini dibuat untuk dilaksanakan dengan penuh tanggung jawab.

Malang, 28 April 2016
Ketua Jurusan,

Ir. Moh. Sholichin, MT.,Ph.D
NIP. 19670602 199802 1 001

Tembusan :
1. Ketua Prodi Informatika / Ilmu Komputer
2. Dosen Pembimbing
3. Arsip

Lampiran 4 Hasil Wawancara Pakar

No	Pertanyaan	Penjelasan
1.	Adakah pengukuran terhadap kualitas airnya? Apa sajakah kelas pengelompokkannya?	Berdasarkan Peraturan Pemerintah No 82 Tahun 2001, dijelaskan kualitas air menjadi 4 kelas yaitu, memenuhi baku mutu, tercemar ringan, tercemar sedang, dan tercemar berat.
2.	Adakah parameter pengukurnya?	Ada. untuk pengambilan parameter memiliki keterbatasan alat, waktu, dan biaya sehingga parameter pengukurnya terbatas.
3.	Parameter apa sajakah yang sangat berpengaruh terhadap kualitas air?	Residu Tersuspensi, BOD, COD, Oksigen Terlarut, derajat keasaman (pH), Fenol, Minyak dan Lemak
4.	Metode apa sajakah yang digunakan dalam pengukuran kualitas air yang dianjurkan pemerintah?	Berdasarkan Keputusan Pemerintah No 115 Tahun 2003 ditetapkan 2 metode untuk pengukuran kualitas air yaitu, metode STORET dan Inpdex Pencemaran.
5.	Adakah kelemahan dari metode tersebut?	Ada.
6.	Apa sajakah kelemahannya?	Perhitungan terlalu sederhana, pemberian skor dan rentang nilai pada parameter kurang menggambarkan kondisi dilapangan, karena setiap daerah memiliki tingkat cemar yang berbeda – beda.

Mengetahui,

Dosen Pakar, Teknik Pengairan

Fakultas Teknik, Universitas Brawijaya

Dr. Eng. Riyanto Haribowo, ST., MT.

NIP: 19770424 200312 1 001

Lampiran 5 Hasil Pengujian

Berikut hasil pengujian dengan nilai akurasi dari 2 skenario pengujian yaitu pengujian nilai *k* dan pengujian *hidden layer* diperoleh akurasi sebesar 87.89% dengan nilai *k* = 5 dan *hidden layer* = 7. Tabel dibawah ini merupakan hasil perolehan optimal dari data pembagian nilai *k* = 5, bobot awal dan bias menggunakan *hidden layer* = 7 dengan bobot *output* hasil proses *training* pada setiap iterasi percobaan *fold* sebagai data *testing*.

Pembentukan bobot awal dan bias menggunakan *hidden layer* = 7

Bobot Awal

No	Bobot 1	Bobot 2	Bobot 3	Bobot 4	Bobot 5	Bobot 6	Bobot 7
1	0.41837447	0.20908803	0.22626809	0.49721349	0.55112715	0.34749858	0.217688
2	0.85167312	0.70940321	0.11936681	0.53421773	0.49998322	0.04648828	0.0188015
3	0.28255657	0.51546371	0.74890819	0.93722205	0.57182635	0.89614683	0.20463954
4	0.63546863	0.23287079	0.95773155	0.25607255	0.4533287	0.96852072	0.02720598
5	0.86325284	0.93977406	0.01811162	0.9217182	0.72429422	0.57240853	0.96427043
6	0.58180052	0.29635008	0.77972134	0.88338758	0.09586649	0.03320766	0.25720456
7	0.84270891	0.39499392	0.70957376	0.8527505	0.76945448	0.77601125	0.20748438

Bias

No	Bias
1	0.98000896
2	0.52893635
3	0.7383708
4	0.40163712
5	0.69725196
6	0.42007313
7	0.25627679

Nilai K = 5

Data Fold 1

No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Kelas
88	0.095735422	0.033695567	0.039584757	0.098064516	0.387755102	0.00026143 8	0.043795562	Ringan
67	0.105744125	0.02509577	0.020850323	0.149677419	0.020408163	0.00026143 8	0.145985401	Ringan
98	0.082463	0.10562	0.045094	0.1251	0.1632	0	0.0583	Ringan

	011	114	884	6129	65306		94161	
80	0.152306353	0.00555 0778	0.050605012	0.2193 54839	0.3877 55102	0.211 7647 06	0.0729 92701	Ringan
13	0.152306353	0.01102 3376	0.052809063	0.1393 54839	0.3469 38776	0.491 5032 68	0.1167 88321	Ringan
55	0.087684943	0.00652 8028	0	0.1122 58065	0.5306 12245	0	0.4160 58394	Ringan
86	0.100087032	0.01297 7875	0.513455732	0.1677 41935	0.3061 22449	0.000 2614 38	0	Ringan
97	0.060922541	0.05558 5959	0.061074254	0.1135 48387	0.1020 40816	0	0.1167 88321	Ringan
73	0.043516101	0.01102 3376	0.386171784	0.0696 77419	0.3265 30612	0	0.1167 88321	Ringan
71	0.087032202	0.01375 9675	0.319499 24	0.2451 6129	0.1632 65306	0	0.1313 86861	Ringan
12	0.110313316	0.02353 2171	0.115073505	0.0838 70968	0.1836 73469	0.394 7712 42	0.5109 48905	Sedang
61	0.057876414	0.05558 5959	0.101298186	0.0954 83871	0.1836 73469	0.000 2614 38	0.0437 9562	Sedang
22	0.111183638	0.01375 9675	0.085318816	0.1174 19355	0.1836 73469	0.044 4444 44	0.0583 94161	Sedang
42	0.189947781	0.05363 146	0.513455732	0.2219 35484	1	0.211 7647 06	0	Sedang
45	0.071801567	0.01297 7875	0.061074254	0.0593 54839	0.3673 46939	0.192 1568 63	0.5255 47445	Sedang
49	0.099651871	0.02587 757	0.107359326	0.1458 06452	0.5714 28571	0.018 3006 54	0.3941 60584	Sedang
48	0.030678851	0.02431 3971	0.073747548	0.0748 3871	0.4285 71429	0.094 1176 47	0.4379 56204	Sedang
28	0.773063534	0.01024 1576	0.062176 28	0.0193 54839	0	0.411 7647 06	0.2189 78102	Sedang
26	0.120104439	0.04737 7062	0.159154526	0.1729 03226	0.4489 79592	0.254 9019 61	0.1167 88321	Sedang
32	0.059399478	0.02900 4769	0.031870578	0.0722 58065	0.3061 22449	0.120 2614 38	0	Sedang
128	0.413402959	0.02900 4769	0.645698794	0.6838 70968	0.5714 28571	0.239 2156 86	0.5474 45255	Berat
143	0.770234987	0.03838 6365	0.502435476	0.0903 22581	0.4285 71429	0.873 2026	0.1167 88321	Berat

						14		
150	0.239338555	0.140020327	0.127746799	0.4	0.183673469	0.2274509 8	0	Berat
46	0.196040035	0.024704871	0.217010866	0.099354839	0.510204082	0.554248366	0.416058394	Berat
50	0.887293299	0.028222969	0.074904675	0.144516129	0.265306122	0.020915033	0.620437956	Berat
58	0.184073107	0.020795872	0.03669194	1	0.224489796	0	0.693430657	Berat
135	0.530896432	0.107028379	0.799982368	0.993548387	0.183673469	0.044444444	0.121167883	Berat
120	0.291557876	0.073567352	0.535496242	0.967741935	0.306122449	0.175163399	0.394160584	Berat
144	0.417754569	0.108748339	0.568557008	0.193548387	0.387755102	0.329411765	0.116788321	Berat
139	0.670147955	0.111875537	0.965286196	0.245161 29	0.448979592	0.254901961	0	Berat

Bobot Output

No	β
1	30.70799
2	-5.6761
3	-8.57063
4	-5.22736
5	-13.6178
6	-0.49306
7	5.108825

Data Fold 2

No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Kelas
95	0.139252	0.017864	0.042891	0.117419	0.306122	0.000261	0.131387	Ringan
70	0.073977	0.011023	0.312887	0.101935	0.265306	0	0	Ringan
68	0.098782	0.014932	0.00983	0.263226	0.142857	0.000261	0	Ringan
83	0.12228	0.007896	0.207093	0.171613	0.632653	0	0.121168	Ringan
75	0.105527	0.006528	0.235778	0.237419	0.510204	0.000261	0.058394	Ringan

72	0.19517	0.010242	0.622005	0.072258	0.183673	0	0.116788	Ringan
89	0.103133	0.024314	0.006524	0.114839	0.306122	0.000261	0.145985	Ringan
94	0.111184	0.020796	0.048952	0.105806	0.77551	0	0	Ringan
77	0.092472	0.105621	0.073748	0.054194	0.408163	0.000261	0.072993	Ringan
78	0.077241	0.014151	0.107359	0.100645	0	0.101961	0.116788	Ringan
7	0.105527	0.045813	0.101298	0.189677	0.306122	0.175163	0.116788	Sedang
60	0.152742	0.019232	0.137268	0.090323	0.244898	0	0.364964	Sedang
54	0.056571	0.017473	0.05779	0.139355	0.653061	0	0.576642	Sedang
47	0.12228	0.033696	0.122237	0.122581	0.693878	0.206536	0.474453	Sedang
64	0.070931	0.068095	0.151991	0.036129	0.22449	0.004967	0.121168	Sedang
24	0.077241	0.041123	0.045095	0.113548	0.367347	0.231373	0	Sedang
65	0.10879	0.099367	0.207093	0.064516	0.346939	0.000261	0.145985	Sedang
5	0.043516	0.03995	0.082564	0.171613	0.408163	0.449673	0.218978	Sedang
9	0.075718	0.045423	0.109012	0.114839	0.102041	0.334641	0.131387	Sedang
17	0.103133	0.086467	0.28754	0.218065	0.673469	0.020915	0	Sedang
129	0.139252	1	0.755901	0.812903	0.795918	0.311111	0.693431	Berat
118	0.774587	0.026659	0.899165	0.890323	0.408163	0.449673	0.474453	Berat
123	0.4047	0.051677	0.866104	0.477419	0.163265	0.24183	0.510949	Berat
16	0.302872	0.158002	1	0.250323	0.795918	0.311111	0	Berat
149	0.565709	0.077476	0.182848	0.683871	0.346939	0.075817	0.116788	Berat
125	0.748477	0.107185	0.822023	0.812903	0.183673	0.394771	0.576642	Berat
142	0	0.047768	0.623658	0.4	0.306122	1	0.218978	Berat
145	0.139252	0.112657	0.116727	0.193548	0.306122	0.120261	0.058394	Berat
133	0.591819	0.057931	0.491415	0.709677	0.265306	0.488889	0.145985	Berat
141	0.696258	0.107967	0.733861	0.735484	0	0.411765	0	Berat

Bobot Output

No	β
1	27.4023
2	-6.7701
3	-5.8054
4	-2.7649
5	-11.5219
6	-0.2586
7	3.10521

Data Fold 3

No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Kelas
105	0	0.020014	0.085319	0.064516	0.673469	0	0.058394	Ringan
92	0.06832	0.014932	0	0.060645	0.632653	0	0.145985	Ringan
96	0.117493	0.019232	0.02636	0.131613	0.183673	0.000261	0.116788	Ringan
102	0.030679	0.047377	0.037932	0.122581	0.163265	0.000261	0	Ringan
74	0.02698	0.011023	1	0.064516	0.367347	0	0.058394	Ringan
103	0.099652	0.104058	0.145379	0.397419	0.571429	0.000261	0.072993	Ringan
85	0.099652	0.010632	0.213705	0.189677	0	0.000261	0.145985	Ringan
107	0.078764	0.026659	0.101298	0.105806	0.326531	0.000261	0.116788	Ringan
112	0.036118	0.145884	0.115074	0.087742	0.102041	0	0.072993	Ringan
109	0.051784	0.111485	0.109012	0.100645	0.77551	0.000261	0.145985	Ringan
30	0.034813	0.022359	0.029116	0.139355	0.428571	0.873203	0	Sedang
51	0.118799	0.036823	0.235778	0.104516	0.612245	0	0.583942	Sedang
56	0.167537	0.015519	0.071847	0.072258	0.22449	0	0.277372	Sedang
52	0.2104	0.026659	0.293794	0.108387	0.55102	0	0.510949	Sedang
53	0.134247	0.030568	0.108764	0.178065	0.897959	0.007059	0.664234	Sedang
6	0.02698	0.079822	0.316193	0.346452	0.77551	0.224837	0.208029	Sedang
25	0.139687	0.029005	0.159155	0.125161	0.510204	0.394771	0.072993	Sedang
66	0.073977	0.104058	0.175134	0.018065	0.102041	0.286013	0.182482	Sedang
63	0.091	0.0610	0.1365	0.0361	0.1836	0.000	0.121168	Sedang

	384	59	63	29	73	261		
15	0.179721	0.091549	0.386172	0.16	0.571429	0.239216	0	Sedang
121	0.361184	0.044641	0.855084	0.735484	0.183673	0.512418	0.620438	Berat
140	0.174064	0.029787	0.160808	0.658065	0.408163	0.261438	0.175182	Berat
146	0.208877	0.041123	0.645699	0.322581	0.408163	0.248366	0.058394	Berat
38	0.380113	0.020405	0.213705	0.100645	0.632653	0.320261	0.423358	Berat
41	0.313316	0.021187	0.048952	0.050323	0.367347	0.075817	0.919708	Berat
27	0.236292	0.104058	0.230235	0.19871	0.408163	0.261438	0.437956	Berat
124	0.282855	0.046204	0.965286	0.270968	0.142857	0.231373	0.664234	Berat
37	0.291123	0.032914	0.047023	0.054194	0.183673	0.227451	0.59854	Berat
148	0.35248	0.14862	0.811003	0.064516	0.673469	0.179085	0.072993	Berat
116	0.078329	0.043859	0.25999	0.812903	0.938776	0.309804	0.525547	Berat

Bobot Output

No	β
1	31.4023
2	-7.4529
3	4.5712
4	-1.0624
5	-9.8152
6	-0.8714
7	5.05216

Data Fold 4

No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Kelas
93	0.067668	0.014932	0.235778	0.082581	0.408163	0	0.116788	Ringan
81	0.179721	0.104058	0.050605	0.388387	0.612245	0.000261	0.116788	Ringan
91	0.030896	0.011023	0.05779	0.085161	0.938776	0	0.121168	Ringan
101	0.12228	0.029005	0.04234	0.219355	0.346939	0.000261	0.131387	Ringan
111	0.117058	0.089985	0.156399	0.019355	0.183673	0	0	Ringan
79	0.110313	0.005551	0.074905	0.175484	0.306122	0.075817	0	Ringan

100	0.19604	0.041123	0.159155	0.19871	0.183673	0.000261	0.122628	Ringan
76	0.075718	0.055586	0.122237	0.105806	0.44898	0.000261	0	Ringan
69	0.082681	0.014932	0.555333	0.114839	0.326531	0	0.116788	Ringan
87	0.130548	0.024705	0.020299	0.139355	0.428571	0.000261	0.145985	Ringan
44	0.082463	0.010632	0.02636	0.064516	0.367347	0.162092	0.510949	Sedang
43	0.060923	0.029396	0.042891	0.055484	0.428571	0.332026	0.875912	Sedang
10	0.092472	0.040341	0.077054	0.101935	0.163265	0.24183	0.116788	Sedang
33	0.117058	0.031741	0.05887	0.069677	0.408163	0.248366	0.510949	Sedang
39	0.367711	0.011023	0.020299	0.175484	0.693878	0.151634	0.729927	Sedang
57	0.075065	0.048354	0.227144	0.136774	0.142857	0	0.547445	Sedang
8	0.11423	0.053241	0.160808	0.212903	0.183673	0.512418	0	Sedang
59	0.111184	0.017864	0.069097	0.051613	0.285714	0	0.729927	Sedang
19	0.030896	0.019623	0.037932	0.060645	0.326531	0.271895	0.189781	Sedang
1	0	0.111485	0.555333	0.19871	0.387755	0.202614	0.116788	Sedang
132	0.195822	0.038386	0.866104	0.451613	0.326531	0.271895	0.043796	Berat
119	0.500435	0.118521	0.744881	0.735484	0.77551	0.224837	0.437956	Berat
136	0.265448	0.077789	0.160808	0.374194	0.326531	0.271895	0.145985	Berat
115	0.509138	0.057931	0.447334	0.167742	0.612245	0.037908	0.510949	Berat
127	0.835509	0.052146	0.25999	0.477419	0.163265	0.296732	0.277372	Berat
35	0.112272	0.040341	0.186705	0.237419	0.673469	0.179085	0.416058	Berat
134	0.021758	0.062231	0.326111	0.812903	0.163265	0.445752	0.121168	Berat
131	0.409051	0.085294	0.921205	0.425806	0.367347	0.460131	0.364964	Berat
137	0.509138	0.154796	0.755901	0.503226	0.367347	0.231373	0.182482	Berat
117	0.004352	0.097021	0.436314	0.812903	0.632653	0.449673	0.416058	Berat

Bobot Output

No	β
1	29.8713

2	-6.9149
3	-7.06715
4	-1.7319
5	-11.8914
6	-1.7513
7	5.4124

Data Fold 5

No	TSS	BOD	COD	DO	pH	Fenol	Minyak dan Lemak	Kelas
40	0.146214	0.011023	0.025809	0.068387	0.408163	0.101961	0.056934	Ringan
82	0.080505	0	0.207093	0.179355	0.938776	0.000261	0.043796	Ringan
84	0.030679	0.030568	0.017544	0.346452	0.408163	0	0.121168	Ringan
90	0.077459	0.011023	0.108764	0.101935	0.612245	0.000261	0.121168	Ringan
99	0.071802	0.061059	0.159155	0.172903	0.142857	0.000261	0.065693	Ringan
104	0	0.010242	0.21635	0.069677	0.795918	0.000261	0.116788	Ringan
106	0.069626	0.022359	0.03132	0.237419	0.367347	0	0.072993	Ringan
108	0.034813	0.023532	0.164114	0.054194	0.265306	0.000261	0.145985	Ringan
110	0.059399	0.069658	0.077054	0.219355	0.306122	0	0.116788	Ringan
113	0.112272	0.03995	0.052809	0.139355	0.163265	0	0.116788	Ringan
11	0.077241	0.030959	0.156399	0	0.142857	0.231373	0.116788	Sedang
14	1	0.021578	0.157501	0.095484	0.163265	0.296732	0	Sedang
18	0.077459	0.01806	0.04234	0.085161	0.367347	0.460131	0	Sedang
20	0.06832	0.037214	0.145379	0.082581	0.265306	0.488889	0.423358	Sedang
21	0.067668	0.016887	0.03132	0.105806	0.163265	0.445752	0.058394	Sedang
29	0.078764	0.020014	0.038483	0.087742	0.306122	1	1	Sedang
31	0.051784	0.026659	0.037381	0.108387	0.387755	0.329412	0.620438	Sedang
34	0.036118	0.036041	0.124441	0.064516	0.571429	0.329412	0	Sedang
36	0.057224	0.041123	0.163563	0.105806	0.346939	0.075817	0.328467	Sedang
62	0.090078	0.105621	0.163012	0.077419	0.22449	0.000261	0.145985	Sedang

2	0	0.069658	0.312887	0.304516	0.612245	0.037908	0.656934	Berat
3	0.19517	0.089985	0.319499	0.388387	0.938776	0.309804	0.620438	Berat
4	0.31658	0.145884	0.622005	0.179355	0.632653	0.449673	0	Berat
23	0.238903	0.051677	0.447334	0.131613	0.326531	0.271895	0.456204	Berat
114	0.6745	0.094285	0.414273	0.658065	0.387755	0.202614	0.875912	Berat
122	0.787641	0.149011	0.24897	0.941935	0.102041	0.334641	0.583942	Berat
126	0.809399	0.119694	0.127747	0.941935	0.346939	0.491503	0.416058	Berat
130	0.644038	0.046204	0.612638	0.606452	0.673469	0.020915	0.729927	Berat
138	0.278503	0.07302	0.083666	0.477419	0.510204	0.394771	0.145985	Berat
147	0.452567	0.161129	0.877124	0.4	0.571429	0.329412	0	Berat

Bobot Output

No	β
1	32.3025
2	-7.2703
3	-5.4127
4	-3.81031
5	-10.2054
6	-0.60815
7	7.30128

Lampiran 6 Tabel Pengambilan Data *Training* Dan Data *Testing*

Berikut ini tabel hasil pengambilan data untuk setiap *fold* dengan nilai *k* = 3 pada proses manualisasi dimana *fold* 3 sebagai data *testing* dan *fold* 1 dan *fold* 2 sebagai data *training*.

Data *Training*

No	TSS	BOD	COD	DO	pH	Fenol	Lemak dan Minyak	Kelas
13	0.152306	0.029677	0.052809063	0.139354839	0.346939	0.491503	0.1167883	1
1	0	0.692903	0.555332702	0.198709677	0.387755	0.202614	0.1167883	2
5	0.043516	0.220645	0.082563752	0.171612903	0.408163	0.449673	0.2189781	2
6	0.02698	0.483871	0.316193163	0.346451613	0.77551	0.224837	0.2080292	2
7	0.105527	0.259355	0.101298186	0.189677419	0.306122	0.175163	0.1167883	2
8	0.11423	0.308387	0.160807564	0.212903226	0.183673	0.512418	0	2
9	0.075718	0.256774	0.109012365	0.11483871	0.102041	0.334641	0.1313869	2
10	0.092472	0.223226	0.077053625	0.101935484	0.163265	0.24183	0.1167883	2
11	0.077241	0.16129	0.156399462	0	0.142857	0.231373	0.1167883	2
12	0.110313	0.112258	0.115073505	0.083870968	0.183673	0.394771	0.5109489	2
14	1	0.099355	0.157501488	0.095483871	0.163265	0.296732	0	2
15	0.179721	0.56129	0.386171784	0.16	0.571429	0.239216	0	2
17	0.103133	0.527742	0.287540499	0.218064516	0.673469	0.020915	0	2
18	0.077459	0.076129	0.042339821	0.08516129	0.367347	0.460131	0	2
19	0.030896	0.086452	0.037931718	0.060645161	0.326531	0.271895	0.189781	2
2	0	0.416774	0.312887086	0.304516129	0.612245	0.037908	0.6569343	3
3	0.19517	0.550968	0.31949924	0.388387097	0.938776	0.309804	0.620438	3
4	0.31658	0.92	0.622005246	0.179354839	0.632653	0.449673	0	3
16	0.302872	1	1	0.250322581	0.795918	0.311111	0	3
40	0.146214	0.029677	0.025809	0.068387	0.408163	0.101961	0.056934	1
20	0.06832	0.202581	0.145379	0.082581	0.265306	0.488889	0.423358	2
21	0.0676	0.0683	0.03132	0.105806	0.1632	0.4457	0.05839	2

	68	87			65	52	4	
22	0.111184	0.047742	0.085319	0.117419	0.183673	0.044444	0.058394	2
24	0.077241	0.228387	0.045095	0.113548	0.367347	0.231373	0	2
25	0.139687	0.148387	0.159155	0.125161	0.510204	0.394771	0.072993	2
26	0.120104	0.269677	0.159155	0.172903	0.44898	0.254902	0.116788	2
28	0.773064	0.024516	0.062176	0.019355	0	0.411765	0.218978	2
29	0.078764	0.089032	0.038483	0.087742	0.306122	1	1	2
30	0.034813	0.104516	0.029116	0.139355	0.428571	0.873203	0	2
31	0.051784	0.132903	0.037381	0.108387	0.387755	0.329412	0.620438	2
32	0.059399	0.148387	0.031871	0.072258	0.306122	0.120261	0	2
33	0.117058	0.166452	0.05887	0.069677	0.408163	0.248366	0.510949	2
34	0.036118	0.194839	0.124441	0.064516	0.571429	0.329412	0	2
36	0.057224	0.228387	0.163563	0.105806	0.346939	0.075817	0.328467	2
23	0.238903	0.298065	0.447334	0.131613	0.326531	0.271895	0.456204	3
27	0.236292	0.643871	0.230235	0.19871	0.408163	0.261438	0.437956	3
35	0.112272	0.223226	0.186705	0.237419	0.673469	0.179085	0.416058	3
37	0.291123	0.174194	0.047023	0.054194	0.183673	0.227451	0.59854	3

Data *Testing*

No	TSS	BOD	COD	DO	pH	Fenol	Lemak dan Minyak	Kelas
55	0.087685	0	0	0.112258	0.530612	0	0.416058	1
39	0.367711	0.029677	0.020299	0.175484	0.693878	0.151634	0.729927	2
42	0.189948	0.310968	0.513456	0.221935	1	0.211765	0	2
43	0.060923	0.150968	0.042891	0.055484	0.428571	0.332026	0.875912	2
44	0.082463	0.027097	0.02636	0.064516	0.367347	0.162092	0.510949	2
45	0.071802	0.042581	0.061074	0.059355	0.367347	0.192157	0.525547	2
47	0.12228	0.179355	0.122237	0.122581	0.693878	0.206536	0.474453	2

48	0.030679	0.117419	0.073748	0.074839	0.428571	0.094118	0.437956	2
49	0.099652	0.127742	0.107359	0.145806	0.571429	0.018301	0.394161	2
51	0.118799	0.2	0.235778	0.104516	0.612245	0	0.583942	2
52	0.2104	0.132903	0.293794	0.108387	0.55102	0	0.510949	2
53	0.134247	0.15871	0.108764	0.178065	0.897959	0.007059	0.664234	2
54	0.056571	0.072258	0.05779	0.139355	0.653061	0	0.576642	2
56	0.167537	0.059355	0.071847	0.072258	0.2249	0	0.277372	2
57	0.075065	0.276129	0.227144	0.136774	0.142857	0	0.547445	2
59	0.111184	0.074839	0.069097	0.051613	0.285714	0	0.729927	2
60	0.152742	0.083871	0.137268	0.090323	0.244898	0	0.364964	2
38	0.380113	0.091613	0.213705	0.100645	0.632653	0.320261	0.423358	3
41	0.313316	0.096774	0.048952	0.050323	0.367347	0.075817	0.919708	3
46	0.19604	0.12	0.217011	0.099355	0.510204	0.554248	0.416058	3
50	0.887293	0.143226	0.074905	0.144516	0.265306	0.020915	0.620438	3
58	0.184073	0.094194	0.036692	1	0.2249	0	0.693431	3

Lampiran 7 Tabel Fungsi Keluaran *Hidden Layer* (Training)

Berikut ini tabel hasil keluaran *hidden layer* pada proses manualisasi dimana *fold* 1 dan *fold* 2 sebagai data *training*.

H	1	2
1	1.208114	1.723029
2	1.571134	1.755029
3	1.344071	1.84733
4	1.734165	2.145731
5	1.099445	1.490792
6	1.30685	1.674237
7	1.055204	1.428092
8	0.972472	1.365484
9	0.878288	1.269643
10	1.287654	1.708783
11	1.424133	1.473318
12	1.540186	1.816581
13	1.357408	1.699219
14	1.062781	1.610557
15	0.965594	1.48866
16	1.68354	2.041107
17	2.282826	2.654435
18	2.131359	2.222856
19	2.394103	2.352176
20	0.834487	1.332694
21	1.389138	1.833438
22	0.981172	1.471677
23	0.734044	1.138134
24	0.987896	1.458146
25	1.291194	1.779233
26	1.265069	1.683496
27	1.283531	1.448401
28	2.031572	2.611628
29	1.436472	2.071249
30	1.36145	1.87374
31	0.782051	1.255202
32	1.281738	1.753714
33	1.132511	1.680339
34	1.093056	1.49118
35	1.611882	1.833491
36	1.747866	1.975417
37	1.530235	1.99315
38	1.283628	1.618778

Lampiran 8 Tabel Aktifasi Fungsi Keluaran *Hidden Layer* (Training)

Berikut ini tabel hasil aktifasi keluaran *hidden layer* pada proses manualisasi dimana *fold* 1 dan *fold* 2 sebagai data *training*.

H(x)	1	2
1	0.76996515	0.848518543
2	0.8279452	0.852586012
3	0.79315862	0.863813301
4	0.84994441	0.895269146
5	0.75015604	0.816197069
6	0.78698565	0.842139884
7	0.74177301	0.806603828
8	0.72561193	0.796649503
9	0.70646741	0.78068166
10	0.78374979	0.846678416
11	0.80598558	0.813561224
12	0.82349179	0.860155426
13	0.79533816	0.845432686
14	0.74322173	0.833488668
15	0.72424049	0.815876988
16	0.84337274	0.885045902
17	0.90744466	0.9342838
18	0.893914	0.902283296
19	0.91637656	0.913107029
20	0.69730276	0.791285868
21	0.80045456	0.862170826
22	0.72734073	0.813312222
23	0.67569203	0.757336901
24	0.72867212	0.811248967
25	0.78434921	0.85560211
26	0.77989739	0.843366868
27	0.78305016	0.809752244
28	0.88407225	0.931606225
29	0.8079077	0.888077187
30	0.79599529	0.866890436
31	0.68612191	0.778198988
32	0.78274539	0.852420591
33	0.75630198	0.842949377
34	0.74895679	0.816255368
35	0.83367257	0.862177024
36	0.85168345	0.878191727
37	0.82204067	0.880075983
38	0.78306679	0.834626532

Lampiran 9 Tabel Matriks *Moore-Penrose Pseudo Inverse*

Berikut ini tabel matriks *moore-penrose* pada proses manualisasi dimana *fold* 1 dan *fold* 2 sebagai data *training*.

H^+	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
1	-0.759598	1.200199	-0.43641	0.556838	-0.38636	0.071017	-0.36612	-0.61501	-0.76907	-0.19902	1.721657	0.783945	0.262085	-1.2201	-1.31283	0.663825	1.321096	1.910586	2.35807
2	0.7392429	-1.08748	0.43854	-0.48619	0.390128	-0.03527	0.370911	0.602558	0.745 58	0.216 62	-1.57501	-0.69918	-0.21326	1.167961	1.253757	-0.58629	-1.19718	-1.74786	-2.1646

H^+	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**
1	-1.45802	-0.11717	-1.11424	-1.09507	-0.99653	-0.47819	-0.2267	1.020581	0.566072	-0.72101	-0.43756	-1.42119	-0.42893	-1.06599	-0.43169	1.083992	1.195 47	0.059875	0.182587
2	1.388196	0.140891	1.068541	1.048614	0.958739	0.477182	0.242304	-0.92162	-0.49346	0.704725	0.439723	1.353381	0.431 15	1.024649	0.432393	-0.9788	-1.08213	-0.02349	-0.13955

Lampiran 10 Tabel Fungsi Keluaran *Hidden Layer* (Testing)

Berikut ini tabel hasil keluaran *hidden layer* pada proses manualisasi dimana *fold* 3 sebagai data *testing*.

H	1	2
1	0.998678	1.543213
2	1.609416	2.083078
3	1.730107	2.136278
4	1.509341	2.023789
5	1.08295	1.591887
6	1.130816	1.632089
7	1.457945	1.974737
8	1.070204	1.567991
9	1.174826	1.654021
10	1.375246	1.791233
11	1.351368	1.710525
12	1.550186	2.091589
13	1.224992	1.771018
14	0.843453	1.232629
15	1.147812	1.421666
16	1.096209	1.513412
17	0.956924	1.323315
18	1.613702	2.020687
19	1.417871	1.787579
20	1.639601	2.109365
21	1.57718	1.656342
22	1.794394	2.05229

Lampiran 11 Tabel Aktifasi Fungsi Keluaran *Hidden Layer* (Testing)

H(x)	1	2
1	0.73079855	0.823931374
2	0.83333026	0.889247571
3	0.84942607	0.894379503
4	0.8189635	0.883272247
5	0.74705178	0.830881475
6	0.75598941	0.836455613
7	0.81121816	0.878118972
8	0.74463578	0.827497085
9	0.76401631	0.839433768
10	0.79822639	0.857078349
11	0.79435316	0.846904429
12	0.82494062	0.890082965
13	0.77294093	0.854584257
14	0.69919202	0.774278423
15	0.75911104	0.805599529
16	0.74954904	0.819566338
17	0.72250556	0.789732686
18	0.83392471	0.882952041
19	0.8050045	0.856630242
20	0.8374806	0.891810058
21	0.8288047	0.839746367
22	0.85746509	0.886178768