TESIS – SM 142501

INTEGRASI SELEKSI DATA DAN *EXTREME LEARNING MACHINE* (ELM) UNTUK PREDIKSI *BINDING SITE* PROTEIN-LIGAN

UMI MAHDIYAH
1213201032

DOSEN PEMBIMBING:
Prof. Dr. M. Isa Irawan, M.T.
Dr. Elly Matul Imah, S.Si., M.Kom.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS - SM 142501

INTEGRATING DATA SELECTION AND EXTREME *LEARNING* MACHINE FOR PREDICTING PROTEIN-LIGAND *BINDING SITE*

UMI MAHDIYAH
1213201032

SUPERVISOR:
Prof. Dr. M. Isa Irawan, MT
Dr. Elly Matul Imah, S.Si., M.Kom.

MAGISTER PROGRAM
DEPARTMENT OF MATHEMATICS
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

INTEGRASI SELEKSI DATA DAN *EXTREME LEARNING MACHINE* (ELM) UNTUK PREDIKSI *BINDING SITE* PROTEIN-LIGAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
UMI MAHDIYAH
NRP. 1213 201 032

Tanggal Ujian : 9 Juli 2015
Periode Wisuda : September 2015

Disetujui oleh:

Prof. Dr. Mohammad Isa Irawan, M.T. (Pembimbing I)
NIP. 19631225 198903 1 001

Dr. Elly Matul Imah, S.Si., M. Kom. (Pembimbing II)
NIP. 19820405 200501 2 002

Dr. Imam Mukhlash, S.Si., M.T. (Penguji)
NIP. 19700831 199403 1 003

Dr. Budi Setiyono, S.Si., M.T. (Penguji)
NIP. 19720207 199702 1 001

Dr. Dwi Ratna Sulistyaningrum, S.Si., M.T. (Penguji)
NIP. 19690405 199403 2 003

Direktur Program Pascasarjana,

Prof. Dr. Ir. Adi Soeprijanto, M.T.
NIP. 19640405 199002 1 001

INTEGRASI SELEKSI DATA DAN *EXTREME LEARNING MACHINE* (ELM) UNTUK PREDIKSI *BINDING SITE* PROTEIN-LIGAN

Nama Mahasiswa	: Umi Mahdiyah
NRP	: 1213 201 032
Pembimbing I	: Prof. Dr. Mohammad Isa Irawan, MT
Pembimbing II	: Dr. Elly Matul Imah, M.Kom.

ABSTRAK

Computer-aided drug design atau desain obat berbantuan komputer mulai banyak dikembangkan saat ini. Langkah awal sebelum melakukan desain obat adalah mencari *binding site* protein – ligan. *Binding site* adalah suatu kantung atau lubang pada permukaan protein yang digunakan untuk menambatkan suatu senyawa ligan (obat). Dalam penelitian ini, prediksi *binding site* dirumuskan sebagai masalah klasifikasi biner, yaitu untuk membedakan lokasi yang bisa mengikat suatu ligan dan lokasi yang tidak bisa mengikat ligan. *Extreme Learning Machine* (ELM) dipilih sebagai algoritma klasifikasi pada penelitian ini, karena ELM mempunyai kelebihan pada waktu komputasinya yang cepat. Dataset dalam kasus nyata kebanyakan berupa *imbalanced* dataset, salah satunya adalah pada masalah prediksi *binding site*. *Imbalanced* data dapat diselesaikan dengan beberapa cara, salah satunya adalah dengan seleksi data. Dalam penelitian ini dilakukan integrasi langkah seleksi data dan ELM, dengan tujuan untuk mengatasi masalah inkonsistensi seleksi data dan klasifikasi. Performa dari integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan diukur mengunakan akurasi, *precision, recall*, dan *g-mean*. Hasil dari penelitia ini diperoleh akurasi rata-rata 96%, *recall* rata-rata 91.8472%, dan *G-mean* rata-rata adalah sebesar 94.26%.*CPU Time* yang dibutuhkan untuk training data protein rata-rata adalah 2.78607detik.

Kata Kunci: *Extreme Learning Machine, imbalanced data,* Protein-ligan *binding site*

INTEGRATING DATA SELECTION AND EXTREME LEARNING MACHINE (ELM) FOR PREDICTING PROTEIN-LIGAN BINDING SITE

Name	: Umi Mahdiyah
Student Identity Number	: 1213 201 032
Supervisor	: Prof. Dr. Mohammad Isa Irawan, MT
Co-Supervisor	: Dr. Elly Matul Imah, M.Kom.

ABSTRACT

Computer-aided drug design has been developed at this time. The initial step computer-aided drug design is find a protein - ligan binding site, which is a pocket or cleft on the surface of the protein used to bind a ligan (drug). In this study the binding site is defined as a binary classification problem, which is to distinguish the location that can bind a ligand and a location that cannot bind ligan. Extreme Learning Machine (ELM) is selected for classification in this study, because of the speed ELM has a fast learning process. In the case of real-world datasets are usually imbalanced datasets, one of which is at issue binding site prediction. Imbalanced data can be solved in several ways, one of which is with the selection of data. This study carried out the integration of data selection and classification, that to overcome the problem of inconsistency of data selection and classification. Performance of integrating data selection and Extreme Learning Machine for predicting protein-ligan binding site was measured using accuracy, precision, recall, and g-mean. The average of accuracy, recall, g-mean and CPU time in this research is 96%, 91.8472%, 94.26%, and 2.78607 second respectively.

Keywords: Protein-ligan binding site, Extreme Learning Machine, imbalanced data, integration.

KATA PENGANTAR

Bismillahirrahmanirrahim,

Alhamdulillahirobbil 'alamin, puji syukur penulis haturkan kehadirat Allah SWT yang senantiasa mencurahkan rahmat, taufik dan hidayah-Nya kepada penulis sehingga dengan rahmat-Nya penulis dapat menyelesaikan tesis yang berjudul **"Integrasi Seleksi Data dan *Extreme Learning Machine* (ELM) Untuk Prediksi *Binding Site* Protein-Ligan"** sebagai salah satu syarat kelulusan Program Studi Strata 2 (S-2) Program Magister Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh Nopember Surabaya dengan baik. Semoga sholawat serta salam tetap tercurahkan kepada Nabi Muhammad SAW yang telah membimbing umat-Nya dari zaman jahiliyah menuju zaman yang penuh ilmu.

Dalam penyelesaian tesis ini, penulis banyak melibatkan berbagai pihak, untuk itu patut kiranya peneliti mengucapkan terima kasih teriring dengan do'a *Jazzakumullaahu ahsanal jazaa* yang disampaikan kepada :

1. Prof. Ir. Joni Hermana, M.Sc.ES., Ph.D selaku RektorInstitut Teknologi Sepuluh Nopember Surabaya.
2. Prof. Dr. Ir. Adi Soeprojanto, M.T. selaku Direktur Program Pascasarjana Institut Teknologi Sepuluh Nopember Surabaya.
3. Prof. Dr. Erna Apriliani, M.Si. selaku Ketua Jurusan Matematika Institut Teknologi Sepuluh Nopember Surabaya.
4. Dr. Budi Setiyono, S.Si., M.T., selaku Dosen Wali yang telah memberikan motivasi, arahan, dan bimbingannya.
5. Dr. Subiono, M.S. selaku Koordinator Program Studi Pascasarjana Matematika Institut Teknologi Sepuluh Nopember Surabaya,
6. Prof. Dr. Mohammad Isa Irawan, MT, selaku dosen pembimbing 1 yang telah meluangkan waktu di tengah kesibukannya untuk berdiskusi dan memberi arahan dalam penyusunan tesis ini.
7. Dr. Elly Matul Imah, S.Si., M. Kom., selaku dosen pembimbing 2 yang telah meluangkan waktu di tengah kesibukannya untuk berdiskusi dan memberi arahan dalam penyusunan tesis ini.

8. Seluruh Dosen Jurusan Matematika Institut Teknologi Sepuluh Nopember Surabaya, terima kasih atas seluruh ilmu, nasihat, dan bimbingannya, serta seluruh Staff Administrasi, terima kasih atas segala bantuannya.
9. Ibunda Siti Juwariyah, Ayahanda Abdul Karim, saudara-saudaraku Moh. Taqwimuddin, Khusnul Azimatul M., Siti Zakiah, serta calon pendamping hidup yang senantiasa memberikan do'a, dukungan dan restunya kepada penulis dalam menuntut ilmu.
10. Teman-teman Pascasarjana Matematika ITS angkatan 2013, terima kasih atas kenangan yang kalian berikan.
11. Semua pihak yang tidak dapat penulis sebutkan satu persatu, penulis hanya mengucapkan terima kasih yang tak terhingga dalam membantu menyelesaikan tesis ini.

Kemudian tak lupa juga bahwa dalam penulisan ini penulis sadari masih ada kekurangan-kekurangan, untuk itu segala masukan dan saran dari semua pihak demi peningkatan dan perbaikan skripsi ini sangat diharapkan. Harapan terakhir penulis adalah semoga tulisan ini dapat memberikan manfaat kepada semua pihak pada umumnya, dan khususnya bagi penulis.

Surabaya, 30 Juli 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Pada bab ini diberikan ulasan mengenai hal-hal yang melatarbelakangi penelitian. Di dalamnya mencakup identifikasi permasalahan, maupun beberapa informasi tentang penelitian terdahulu yang berhubungan dengan topik tesis. Uraian ini bersifat umum dan menjelaskan secara ringkas hal-hal yang dilakukan pada proses penelitian. Dari informasi tersebut kemudian dirumuskan permasalahan yang dibahas, batasan masalah, tujuan, dan manfaat penelitian.

1.1 Latar Belakang

Bioinformatika merupakan ilmu multidisipliner yang melibatkan berbagai bidang ilmu, yang meliputi biologi molekuler, matematika, ilmu komputasi, kimia molekul, fisika, dan beberapa disiplin ilmu lainnya (Shen, 2008). Selama ini, bioinformatika banyak diaplikasikan dalam berbagai masalah, salah satunya adalah masalah desain obat. Konsep desain obat pada bioinformatika didasarkan pada fungsionalitas dari protein, yaitu pencarian sebuah senyawa untuk mengaktifkan atau menghambat fungsi protein target, sebab protein dapat mengekspresikan suatu informasi genetik.

Desain obat secara umum ada dua macam, yaitu desain obat berbasis ligan dan desain obat berbasis struktur. Desain obat bersasis ligan adalah desain obat berbantuan komputer yang didasarkan dari informasi dari ligan (senyawa kimia). Desain obat dilakukan dengan mencocokan satu ligan dengan beberapa protein, sehingga dapat diketahui obat tersebut dapat mengobati suatu penyakit tertentu. Sedangkan desain obat berbasis struktur adalah desain obat yang didasarkan pada striktur kimia dan geometri dari protein. Desain obat dilakukan berdasarkan informasi dari struktur protein untuk mencari ligan yang cocok (Wong, 2013). Dalam penelitian ini pencarian *binding site* dilakukan berdasarkan informasi dari struktur protein, yaitu dari hasil analisis geometri, *sequence* dan energi dari protein.

Desain obat secara umum ada dua macam, yaitu desain obat berbasis ligan dan desain obat berbasis struktur. Desain obat berbasis ligan adalah desain

obat berbantuan komputer yang didasarkan dari informasi dari ligan (senyawa kimia) (Wong, 2013). Desain obat dilakukan dengan mencocokan satu ligan dengan beberapa protein, sehingga dapat diketahui obat tersebut dapat mengobati suatu penyakit tertentu. Sedangkan desain obat berbasis struktur adalah desain obat yang didasarkan pada striktur kimia dan geometri dari protein. Desain obat dilakukan berdasarkan informasi dari struktur protein untuk mencari ligan yang cocok. Dalam penelitian ini pencarian *binding site* dilakukan berdasarkan informasi dari struktur protein, yaitu dari hasil analisis geometri, *sequence* dan energi dari protein.

Beberapa aplikasi bioinformatika dalam desain obat yang sudah banyak dikenal adalah *virtual screening*, *docking*, dan *de novo drug design*. Sebelum melakukan implementasi dengan aplikasi bioinformatika tersebut, struktur tiga dimensi dari target dan *binding site* protein-ligan harus ditemukan telebih dahulu. *Binding site* adalah suatu lubang atau kantung yang terdapat pada permukaan protein yang nantinya akan digunakan untuk menambatkan suatu ligan (senyawa obat).

Saat ini mulai banyak pendekatan komputasi berbasis struktur dan *sequence* yang telah dikembangkan untuk memprediksi *binding site* (Wong, 2012). Pertama, dalam *paper* yang ditulis Capra (2007), yaitu pendekatan berbasis *sequence* dengan menggunakan analisis *sequence conservation* digunakan untuk memprediksi fungsional dari residu penting dalam *sequence* protein (Capra, 2007). Dengan mengetahui residu yang terdapat dalam suatu protein kita dapat pengetahui posisi *binding site* protein-ligan nya. Kedua, dalam *paper* yang ditulis Hendlich dkk. pada tahun 1997 tentang LIGSITE, *paper* yang ditulis oleh Laskowsky tentang SURFNET (Lurie dkk,1995) yang membahas pendekatan berbasis struktur menggunakan analisis geometri ataupun analisis *sequence* untuk menetapkan *binding site*.

Kelemahan dari beberapa pendekatan di atas terletak pada akurasinya. Analisis *sequence conservation* kurang akurat untuk menetukan *binding site* pada suatu protein. Analisis struktur juga kurang akurat, karena hanya *pocket* yang besar saja yang biasanya terdeteksi, selain itu sulit pula digunakan untuk protein

yang *multi-chain*, karena rongga antar *chain* juga dianggap sebagai pocket. (Wong, 2012)

Machine Learning sudah banyak digunakan pada bioinformatik serta menunjukkan hasil yang baik untuk prediksi *binding site*. Dapat dilihat pada beberapa paper yang ditulis oleh Gutteridge dkk. (2003), Koike dkk. (2004), Keil dkk. (2004), Wong dkk. (2013), Wang dkk. (2014). *Extreme Learning Machine* (ELM) adalah salah suatu algoritma *learning* baru pada Jaringan Syaraf Tiruan yang merupakan *Single-hidden layer Feedforward Network* (SLFN). ELM mempunyai kemampuan *learning* yang cepat dan *training error* yang kecil (Huang, 2006).

Prediksi dari *binding site* dapat dirumuskan sebagai masalah klasifikasi biner, yaitu untuk membedakan lokasi *binding site* dan bukan *binding site*. *Extreme Learning Machine* merupakan salah satu *machine learning* untuk pengenalan pola dan klasifikasi dengan performa yang bagus (Huang, 2006). Dalam penelitian ini ELM dipilih karena mempunyai waktu komputasi yang relatif jauh lebih cepat dibandingkan jaringan syaraf tiruan yang lain, selain itu menurut Chorowski (2014) ELM memunyai akurasi yang bagus dan mempunyai akurasi yang tidak jauh berbeda dengan SVM untuk data yang *balanced*.

Sebagaimana seperti data bioinformatika lainnya, data *binding site* protein-ligan merupakan dataset yang mempunyai karakter *imbalanced*. *Imbalanced* data adalah data yang mempunyai perbandingan data positif dan data negatif yang tidak seimbang. *Imbalanced* data merupakan masalah yang cukup besar pada suatu *machine learning*, karena *imbalanced* data dapat mempengaruhi performa suatu *machine learning*.

Ada banyak cara untuk mengatasi masalah *imbalanced* data, diantaranya adalah *undersampling* dan *oversampling*. *Undersampling* adalah cara mengatasi masalah *imbalanced* data dengan cara mengurangi data mayoritas, sehingga diperoleh data dengan proporsi yang tepat atau bahkan data yang *balanced*. *Oversampling* adalah cara mengatasi masalah *imbalanced* dengan melipat gandakan data minoritas menjadi beberapa kali lipat, sehingga diperoleh, sehingga diperoleh data dengan proporsi yang tepat atau bahkan data yang *balance*. Kedua metode tersebut masing masing mempunyai beberapa kelemahan dan kelebihan.

Undersampling untuk mengatasi masalah *imbalanced* data mempunyai kelebihan pada waktu komputasi dan penggunaaan memori, hal tersebut dikarenakan adanya pengurangan data mayoritas, sehingga waktu dan memori yang digunakan menjadi berkurang. Kelemahan dari *undersampling* adalah adanya resiko informasi penting yang terbuang pada saat melakukan undersampling, selain itu jika proses *undersampling* dilakukann secara terpisah dengan proses klasifikasi, beasr kemungkinan terjadi inkonsistensi antara keduanya. Artinya, ketika sudah terlihat optimal pada saat proses *undersampling* (sudah *balance* atau dianggap mempunyai proporsi yang tepat)belum tentu optimal pada saat digunakan pada proses klasifikasi, sehingga perlu melakukan *undersampling* beberapa kali sampai mendapatkan data yang *balance* atau dianggap mempunyai proporsi yang tepat serta bisa mendapatkan hasil yang optimal pula pada saat proses klasifikasi. Sehingga hal tersebut membutuhkan waktu yang relatif lama. *Oversampling* mempunyai kelebihan tidak adanya informasi penting yang terbuang, karena *oversampling* melipat gandakan data minoritas. Kelemahan dari *oversampling*, karena melipat gandakan data, maka memori yang dibutuhkan menjadi lebih besar serta waktu komputasi yang dibutuhkan lebih banyak, sehingga pada saat digunakan pada data dengan ukuran yang besar, maka *cost computation* yang dibutuhkan juga sangat tinggi. Menurut Imah (2014) suatu sistem pengenalan pola mempunyai kelemahan berupa kondisi inkonsistensi antara seleksi data dan klasifikasi ketika kedua langkah tersebut dilakukan secara terpisah, maka perlu dilakukan integrasi pada kedua langkah tersebut.

Berdasarkan kelebihan dan kekurangan dari metode yang sebelumya, maka dalam penelitian ini dilakukan integrasi seleksi data dan ELM. Tujuan melakukan seleksi data supaya tidak memerlukan memori yang besar dan waktu komputasi yang lama. Integrasi dilakukan dengan tujuan untuk mengatasi masalah inkonsistensi antara proses seleksi data dan klasifikasi, karena proses seleksi data dilakukan dalam sistem. Sehingga dalam penelitian ini dilakukan Integrasi Seleksi Data dan *Extreme Learning Machine* (IDELM) untuk prediksi *binding site* protein-ligan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat disusun rumusan masalah sebagai berikut:

a. Bagaimana mengintegrasikan seleksi data dan *Extreme Learning Machine* untuk menangani masalah *imbalanced data*.
b. Bagaimana implementasi integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan.
c. Bagaimana performa dari penerapan integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah penelitian, maka tujuan dari penelitian ini adalah sebagai berikut:

a. Mengintegrasikan seleksi data dan *Extreme Learning Machine* untuk menangani masalah *imbalanced data*,
b. Mengimplementasikan integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan,
c. Mengetahui performa dari penerapan integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan

1.4 Batasan Masalah

Batasan masalah yang dibahas dalam tesis ini adalah sebagai berikut:

a. Dataset yang digunakan adalah data *protein data bank* dari web RCSB Protein Data Bank,
b. Untuk mendapatkan nilai setiap atribut pada protein menggunakan *webserver* LISE
c. Bagian yang diteliti adalah *binding site* protein-ligan dari dataset protein yang digunakan.
d. Implementasi program dengan menggunakan software Matlab.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini sebagai berikut:

a. Integrasi seleksi data dan *Extreme Learning Machine* diharapkan dapat mengatasi masalah *imbalanced* data yang merupakan masalah yang cukup besar pada *machine learning*.
b. Integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi binding site protein-ligan diharapkan dapat digunakan untuk penelitian dasar desain obat berbantuan komputer, karena dengan menemukan *binding site* maka proses desain obat berbantuan komputer lebih terarah.

BAB 2
TINJAUAN PUSTAKA DAN DASAR TEORI

Pada bab ini diberikan tinjauan pustaka dan dasar teori yang digunakan dalam proses penelitian. Beberapa teori yang digunakan meliputi konsep seputar protein, interaksi protein-ligan, *binding site*, prediksi protein ligan *binding site*, Jaringan Syaraf Tiruan, *The Moore–Penrose Generalized Inverse, Minimum Norm Least-Square* dari Bentuk Umum Sistem Linear, *Machine Learning*, *Extreme Learning Machine, Imbalanced Data, Cross Validation*.

2.1 Penelitian Terdahulu

Pencarian *binding site* merupakan suatu permasalahan yang banyak dibahas dalam dunia komputasi. Beberapa penelitian terkait dengan pencarian *binding site* sebelumnya adalah sebagia berikut: (1) Ginny Y. Wong dkk (2012) melakukan prediksi *binding site* dengan menggunakan *Differential Evolution* dan *Support Vector Machine*. Dalam penelitian tersebut dihasilkan *precision* sebesar 59,20% dan *sensitifitas* 59,78% untuk 37 protein. *Precision* sebesar 59,20%, artinya prediksi *binding site* dengan menggunakan *Differential Evolution* dan *Support Vector Machine* dari semua data yang diklasifikasikan menjadi *binding site* hanya 59,20% yang benar. Sedangkan *sensitifitas* 59,78% artinya akurasi data *binding site* nya adalah 59, 78%. (2) Selanjutnya Ginny Y. Wong dkk (2013) melanjutkan penelitian tersebut, yaitu melakukan prediksi *binding site* dengan menggunakan *Support Vector Machine* dengan properti dari protein. Dalam penelitian tersebut dihasilkan *success rate* sebesar 88% dari 198 dataset *drug-target*. S*uccess rate* sebesar 88% dalam penelitian tersebut, artinya 88% dari 198 data protein yang digunakan dapat memprediksi *binding site* lebih dari satu tempat (*binding site*) dalam satu protein.

Penelitian bioinformatika dengan *machine learning* banyak digunakan, diantaranya (1) "*Fast prediction of protein–protein interaction sites based on Extreme Learning Machines*" adalah penelitian yang dilakukan Debby D. Wang dkk (2014). Dalam penelitian tersebut dihasilkan akurasi rata-rata dari 58,5%, *precision* rata–rata 55,6%, dan *recall* rata-rata 56,8% dari 15 data yang

digunakan. (2) "*Protein Sequence Classification Using Extreme Learning Machine*" merupakan penelitian yang dilakukan oleh Dianhui Wang dan Guang-Bin Huang (2013). Tujuan dari klasifikasi *sequence* pada paper tersebut adalah untuk mengklasifikasikan jenis protein berdasarkan kemiripan *sequence* maupun struktur tiga dimensinya. Dalam penelitian Wang digunakan ELM dengan kernel RBF. Hasil penelitian tersebut menunjukkan bahwa ELM dengan menggunakan kernel RBF mendapatkan akurasi sebesar 99.705% dengan waktu komputasi yang dibutuhkan sebesar 9.4351 detik. Pada penelitian ini digunakan integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein-ligan.

2.2 Protein

Protein adalah salah satu bio-makromolekul yang penting peranannya dalam makhluk hidup. Protein tersusun dari 20 macam asam amino alami. Asam amino penyusun protein mengandung beberapa atom kimia seperti carbon(C), nitrogen (N), dan hidrogen (H), kecuali *cyteine* dan *methionine* juga mengandung sulfur (S) (Zvelebil, 2008).

Data tiga dimensi protein adalah hasil dari *X-ray crystallography*, *nuclear magnetic resonance* (NMR*), Spectocropy*, dan *cryoelectron microscopy*. Data tiga dimensi tersebut adalah berupa posisi titik koordinat (x, y, z) setiap atom pada suatu molekul yang berupa data PDB (*Protein Data bank*) atau PQS (*Protein Quartenary Structure*) dengan titik (0, 0, 0) adalah sebagai titik pusat (Zvelebil, 2008). Dari kedua bentuk data tersebut dapat dilihat bentuk visualisasi struktur suatu protein.

Konsep desain obat pada bioinformatika didasarkan pada fungsionalitas dari protein, yaitu pencarian sebuah senyawa untuk mengaktifkan atau menghambat fungsi protein target, sebab protein dapat mengekspresikan suatu informasi genetik. Sebagaimana terdapat ribuan gen di dalam inti sel, masing-masing mencirikan satu sifat nyata dari organisme, di dalam sel terdapat ribuan jenis protein yang berbeda, masing-masing membawa fungsi spesifik yang ditentukan oleh gen yang sesuai (Zvelebil, 2008).

2.3 Interaksi Protein-Ligan

Interaksi protein-ligan merupakan dasar dari proses fundamental secara biologis seperti reaksi enzimatik dan molekul *recognition*. Dalam banyak kasus, interaksi ligan dengan protein melibatkan reorganisasi yang kompleks dari target protein dan ligan. Pola Interaksi protein-ligan berupa ikatan perubahan kualitatif dalam proses pengikatan (Rovira, 2005). Obat (ligan) biasanya terbuat dari seyawa kecil yang mampu mengingat spesifik (atom) pada permukaan atau bagian dalam suatu protein. Ikatan itu mempengaruhi fungsi dari protein, baik menghambat atau memicu aktivitasnya (Widodo, 2014).

Interaksi protein dengan ligan, protein, atau *surfaces* dikendalikan oleh *array* kompleks dari gaya antar molekul dan *intersurface* (Leckband, 2000). Interaksi protein-ligan diperantarai oleh beberapa ikatan hidrogen, interaksi *van der Waals* dan *hidropobic*. Interaksi ligan dengan protein di atas terjadi hanya apabila terdapat kecocokan bentuk dan volume di antara molekul ligan dan situs aktif atau situs tambat protein tersebut (Motiejunas & Wade, 2006). Selain itu, gugus-gugus fungsional pada molekul ligan itu harus berada pada posisi yang memadai dari asam-asam amino yang menjadi pasangannya pada *active site* atau *binding site* tersebut (Schneider & Baringhaus, 2008). Jenis-jenis gugus fungsional ligan dan jarak interaksinya dapat dilihat pada Tabel 2.1.

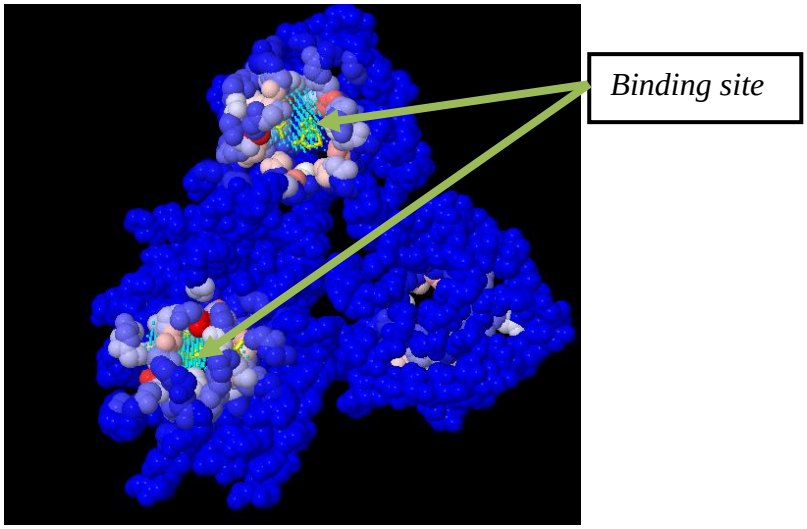
Tabel 2.1 Tabel Gugus Fungsional Ligan dan Jarak Interaksinya

No	Jenis Gugus Fungsional Ligan	Jenis Asam Amino Pasangan	Jarak Interaksi (Å)
1	Bermuatan (+)	Bermuatan (+)	≤ 3.0
2	Bermuatan (-)	Bermuatan (-)	≤ 3.0
3	Donor ikatan Hidrogen	Donor ikatan Hidrogen	3.0 – 3.5
4	Akseptor ikatan Hidrogen	Akseptor ikatan Hidrogen	3.0 – 3.5
5	Aromatik	Aromatik	≥ 6.0
6	Hidrofobik	Hidrofobik	≥ 6.0

2.4 *Binding site*

Binding site protein-ligan adalah bagian permukaan *reseptor* (protein) yang berfungsi untuk melekatnya suatu ligan (obat). *Binding site* protein-ligan umumnya berada di *pocket* (celah, galur) pada permukaan protein

(Hendlich,1997). Penentuan *pocket* merupakan langkah penting menuju desain obat dan penemuan ligan baru. Sebuah analisis mendalam dan klasifikasi *pocket* pada permukaan struktur protein juga dapat meningkatkan pemahaman kita tentang proses yang terlibat dalam pengikatan ligan.



Gambar 2.1 Gambar lokasi yang merupakan *binding site*
(Sumber: webserver Concavity)

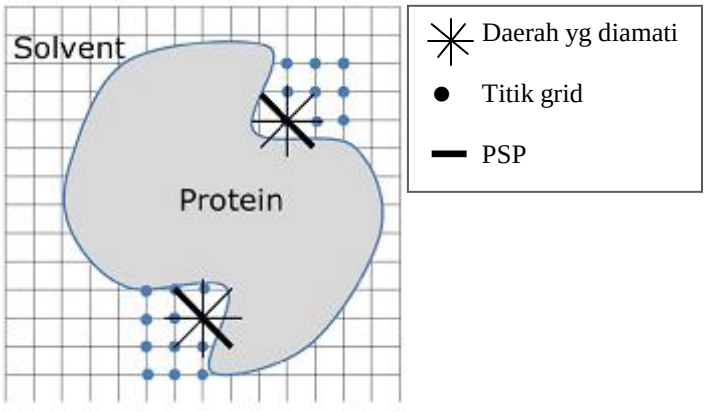
2.5 Prediksi *Binding site* Protein-ligan

Untuk memprediksi suatu *binding site* protein-ligan ada beberapa cara, diantaranya adalah berdasarkan geometri, energy, dan *sequence*.

2.5.1 Prediksi dengan Pendekatan Berdasar Geometri

LIGSITE (Hendlich, 1997) merupakan salah satu metode berbasis geometri untuk menemukan suatu *binding site*. LIGSITE merupakan perbaikan dari algoritma yang dikembangkan oleh Levitt dan Banaszak pada tahun 1992 yaitu program POCKET. Program ini dimulai dengan membuat *cartesian grid* yang teratur. Kedua, pemeriksaan jarak diterapkan pada grid untuk memastikan atom protein tidak tumpang tindih dengan titik grid. Semua titik-titik grid, yang tidak tumpang tindih dengan atom protein, diberi label sebagai pelarut. Jika titik-titik grid luar protein yang tertutup oleh permukaan protein, yaitu titik-titik grid diapit oleh pasangan atom dalam protein, itu disebut protein-solvent-protein (PSP) *event*.

Langkah awal LIGSITE dan POCKET sama seperti yang dijelaskan sebellumnya. Keduanya sama – sama identifikasi PSP *event* berdasarkan koordinat atom. Scan LIGSITE untuk pocket sepanjang tiga sumbu dan empat diagonal ruang sementara POCKET hanya *scaning* pada tiga sumbu saja. Beberapa nilai digunakan untuk setiap titik grid merupakan jumlah PSP *event* yang terdeteksi saat *scanning* pada setiap sumbu. Artinya, semakin tinggi nilai titik grid, semakin besar kemungkinan titik grid menjadi kandidat *binding site*. Metode tersebut hanya berfokus pada karakteristik geometris dan tidak mempertimbangkan karakteristik lain dari protein(Hendlich, 1997). Gambaran dari PSP *event* dapat dilihat pada Gambar 2.1.



Gambar 2.2 PSP event digunakan untuk mendeskripsikan fitur geometri sebuah titik grid (Wong, 2013)

2.5.2. Prediksi dengan Pendekatan Berdasar *Sequence*

Fungsi protein mempunyai peranan penting dalam interaksinya dengan molekul lain. Mengidentifikasi residu berpartisipasi dalam interaksi komponen penting dari fungsional karakteristik protein. Banyak pendekatan komputasi berdasarkan analisis sekuens protein atau struktur telah dikembangkan untuk memprediksi berbagai situs fungsional protein, termasuk *binding site* protein-ligan (Lopez, 2007), *binding site* protein-DNA, *catalytic sites*, interaksi protein-protein *interface* (PPI) dan menentukan posisi spesifisitas.

Selama 15 tahun terakhir, beberapa metode untuk memprediksi *binding site* suatu molekul kecil telah dikembangkan. Pendekatan struktural menggunakan kriteria geometris dan energi untuk menemukan daerah cekung

pada permukaan protein yang cenderung mengikat ligan. Pendekatan berbasis *sequence* yang sering digunakan adalah *sequence coservation* (Valdar, 2002).

Semua residu dalam protein belum tentu selalu penting. Hanya beberapa saja yang penting dalam suatu struktur penting protein dan fungsi protein, sedangkan beberapa yang lain dapat digantikan. Analisis konservasi adalah salah satu dari banyak metode yang digunakan untuk memprediksi fungsional residu penting dari *sequence* protein (Capra, 2007).

2.6 Jaringan Syaraf Tiruan (JST)

Menurut Siang (2009) jaringan syaraf tiruan adalah pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi. Jaringan syaraf tiruan dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi, dengan asumsi beberapa asumsi sebagai berikut:

a. Pemrosesan informasi terjadi pada banyak elemen sederhana(*neuron*).
b. Sinyal dikirimkan diantara *neuron* – *neuron* melalui penghubung.
c. Penghubung antar *neuron* memiliki bobot yang memperkuat atau memperlemah sinyal.
d. Untuk menentukan *output* setiap *neuron* menggunakan fungsi aktivasi yang dikenakan pada jumlah *input* yang diterima. Besarnya *output* selanjutnya dibandingkan dengan *treshold*.

2.7 Machine Learning

Machine Learning adalah sebuah bidang riset yang menggabungkan matematika, statistika, logika inferensi, analisa dan visualisasi data. Menurut Mitchell (1997) *Machine Learning* merupakan daerah penelitian yang diakui ilmu komputer, terutama berkaitan dengan penemuan model, pola, dan keteraturan lainnya dalam data.

Machine Learning adalah salah satu disiplin ilmu dari *Computer Science* yang mempelajari bagaimana membuat komputer/mesin itu mempunyai suatu kecerdasan. Agar mempunyai suatu kecerdasan, komputer/mesin harus dapat belajar. Dengan kata lain, *Machine Learning* adalah suatu bidang keilmuan yang berisi tentang pembelajaran komputer atau mesin untuk menjadi cerdas.

2.8. The Moore–Penrose Generalized Inverse

Pemecahan dari sistem linear umum $Ax = b$, ketika A matriks singular dan bukan matriks persegi, maka dapat diselesaikan dengan menggunakan *Moore–Penrose Generalized Inverse* (Serre, 2002).

Definisi 2.1 Diberikan sebuah matrix $A \in M_{n\times m}(\mathbb{C})$. Terdapat matriks *unique* $A^{\dagger} \in M_{m\times n}(\mathbb{C})$, yang dinamakan *Moore–Penrose Generalized Inverse*, yang memenuhi empat *properties* berikut:

1. $AA^{\dagger}A = A$;
2. $A^{\dagger}AA^{\dagger} = A^{\dagger}$
3. $AA^{\dagger} \in H_n$
4. $A^{\dagger}A \in H_m$

dengan H adalah matriks *Hermitian* (Macausland, 2014)

Ada beberapa metode yang dapat digunakan untuk menghitung *Moore–Penrose Generalized Inverse*, yaitu proyeksi ortogonal, metode *ortogonalization*, metode *iterative*, dan *Singular Value Decomposition* (SVD) (Ortega dalam Huang, 2006).

Teorema 2.1 Jika A matriks singular dan bukan suatu matriks persegi maka $\left(A^T A\right)^{-1} A^T$ merupakan *Moore Penrose Generalized Invers*.

(Macausland, 2014)

Bukti:

Untuk membuktikan bahwa $\left(A^T A\right)^{-1} A^T$ merupakan *Moore Penrose Generalized Invers*, harus dibuktikan bahwa $\left(A^T A\right)^{-1} A^T$ memenuhi empat sifat seperti pada Definisi 2.1. Misalkan $A^{\dagger} = \left(A^T A\right)^{-1} A^T$, dibuktikan $A^{\dagger}$memenuhi empat sifat pada Definisi 2.1.

a. $AA^{\dagger}A = A$

$$A\left(\left(A^T A\right)^{-1} A^T\right)A = A\left(A^T A\right)^{-1} A^T A$$

$$= A\left(A^T A\right)^{-1}\left(A^T A\right)$$

$$= AI = A$$

b. $GA^{\dagger}G = G$

$$\left(\left(A^TA\right)^{-1}A^T\right)A\left(\left(A^TA\right)^{-1}A^T\right)=\left(A^TA\right)^{-1}\left(A^TA\right)\left(\left(A^TA\right)^{-1}A^T\right)$$

$$= I\left(\left(A^TA\right)^{-1}A^T\right)$$

$$=\left(A^TA\right)^{-1}A^T$$

$$= A^{\dagger}$$

c. $AA^{\dagger} \in H_n$

Matriks Hermitian adalah matriks bujur sangkar yang elemen matriksnya adalah bilangan kompleks dan A=A^{T} (Kohno, 2008)

$$\left(AA^{\dagger}\right)^T = AA^{\dagger}$$

$$\left(A\left(\left(A^TA\right)^{-1}A^T\right)\right)^T =\left(\left(A^TA\right)^{-1}A^T\right)^TA^T$$

$$= A\left(\left(A^TA\right)^{-1}A^T\right)$$

$$=AG$$

d. $A^{\dagger}A \in H_m$

$$\left(\left(A^T\left(AA^T\right)^{-1}\right)A\right)^T = A^T\left(A^T\left(AA^T\right)^{-1}\right)^T$$

$$= A^T\left(AA^T\right)^{-1}A$$

$$=\left(A^T\left(AA^T\right)^{-1}\right)A$$

$$= A^{\dagger}A$$

Karena $A^{\dagger}=\left(A^TA\right)^{-1}A^T$ memenuhi empat properties *Moore–Penrose Generalized Inverse*, maka terbukti bahwa $A^{\dagger}$ adalah *Moore–Penrose Generalized Inverse*.

2.9 *Minimum Norm Least-Square* dari Bentuk Umum Sistem Linear

Untuk bentuk umum dari sistem linear $Ax = y$, kita dapat mengatakan $\hat{x}$ adalah solusi kuadrat terkecil (*least square*) jika:

$$\|A\hat{x} - y\| = \min_x \|Ax - y\| \tag{2.1}$$

dengan $\|.\|$ adalah norm pada ruang Euclid

Definisi 2.2 $x_0 \in R^n$ dikatakan solusi *minimum least square* dari sistem linear $Ax = y$ jika setiap $y \in R^m$

$$\|x_0\| \leq \|x\|,$$

$$\forall x \in \|Ax_0 - y\| \leq \|Ax - y\|, \forall x \in R^n$$

(Huang, 2004).

Artinya, sebuah solusi x_0 dikatakan solusi minimum kuadrat terkecil dari sistem linear $Ax = y$ jika x_0 terkecil diantara semua solusi kuadrat terkecil.

Teorema 2.2 Terdapat G sedemikian hingga Gy adalah *minimum norm least square* dari sistem linear $Ax = y$. Sehingga $G = A^{\dagger}$ adalah syarat cukup dan perlu, *Moore–Penrose Generalized Inverse* dari matriks A.

Bukti:

Karena $G = A^{\dagger}$ adalah syarat cukup dan perlu maka harus dibuktikan dua arah

a. Jika Gy adalah *minimum norm least square* dari sistem linear $Ax = y$, maka $G = A^{\dagger}$ adalah *Moore–Penrose Generalized Inverse* dari matriks A.

   Karena Gy adalah *minimum norm least square* dari sistem linear $Ax = y$, sehingga

   $$x = Gy. \tag{2.2}$$

   Jika A adalah matriks persegi, maka solusi dari $Ax = y$ adalah

   $$x = A^{-1}y, \tag{2.3}$$

   karena sifat dari invers adalah tunggal, maka berdasarkan (2.2) dan (2.3) maka

   $$G = A^{-1}.$$

   Jika A bukan matriks persegi, solusi dari $Ax = y$ adalah

   $$Ax = y$$

$$A^T Ax = A^T y$$

$$x = \left(A^T A\right)^{-1} A^T y \tag{2.4}$$

berdasarkan (2) dan (4) maka

$$G = \left(A^T A\right)^{-1} A^T \tag{2.5}$$

Berdasarkan Torema 2.1 *G* adalah *Moore–Penrose Generalized Inverse*.

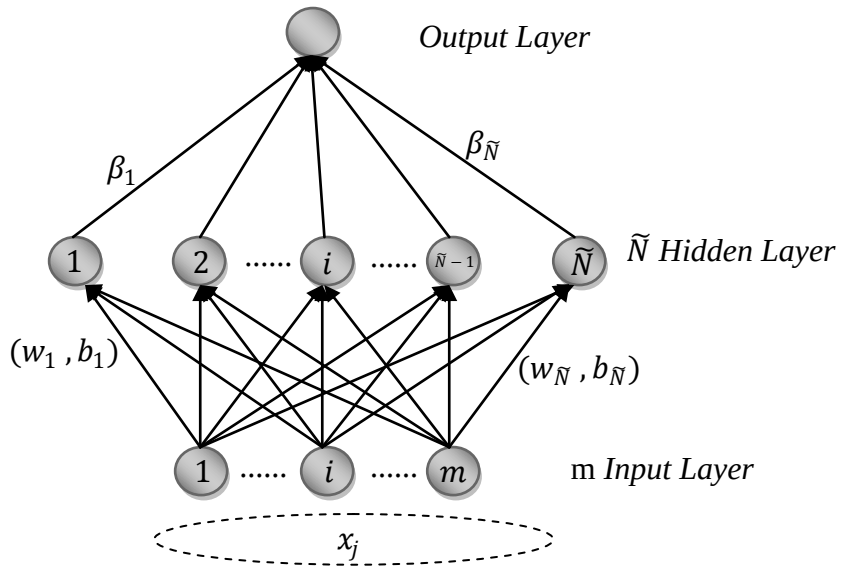
b. Jika $G = A^{\dagger}$ adalah *Moore–Penrose Generalized Inverse* dari matriks A, maka Gy adalah *minimum norm least square* dari sistem linear $Ax = y$.

Berdasarkan Definisi 2.1 dan Teorema 2.1 maka $G = A^{\dagger}$ adalah *Moore–Penrose Generalized Inverse* merupakan penyelesaian dari $Ax = y$.

2.10 *Extreme Learning Machine*(ELM)

Extreme Learning Machine pertama kali di usulkan oleh Huang(2004), yang merupakan algoritma learning sederhana untuk *Single Hidden layer Feedforward Networks* (SLFN) dengan kecepatan learning dapat sangat cepat dibandingkan algoritma *learning feedforward network* biasa.

Arsitektur jaringan *Extreme Learning Machine* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Arsitektur jaringan *Extreme Learning Machine*

Artsitektur jaringan *Extreme Learning Machine* pada Gambar 2.3 menggambarkan bahwa terdapat sebuah data dengan m atribut dari $1,2,\dots,m$, dengan banyak *node* pada *hidden layer* adalah $\tilde{N}$. Setiap *node* pada *input* dihubungkan pada setiap *hidden node* dengan menggunakan fungsi $g_i(w_i x + b_i)$. Hari hasil perhitungan fungsi aktivasi data pertama dengan m atribut ke semua *hidden node* dituliskan pada vektor $h_1(x)$.

$$h_1(x) = [g(w_1 x_1 + b_1) \quad \dots \quad g(w_{\tilde{N}} x_1 + b_{\tilde{N}})]$$

Selanjutnya dicari bobot antara hidden layer dan output layer dengan mengalikan *pseudo invers* dari $h_1(x)$ dikalikan target dari x data pertama, sehingga diperoleh

$$.\beta = h_i(x)^{\dagger} t$$

Setelah itu dilakukan perhitungan output dengan rumus

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i . x_1 + b_i\right) = o_1 .$$

Untuk sebuah himpunan N sampel yang berbeda (x_j, t_j) dengan $x_j = [x_{j1},\ x_{j2}, \dots, x_{jm}]^T \in \mathbb{R}^{n \times m}\ dan\ t_i = [t_1,\ t_2, \dots, t_n]^T \in \mathbb{R}^n$. Maka SLFN standart dengan $\tilde{N}$ *hidden node*s dan fungsi aktivasi $g(x)$ dimodelkan dalam bentuk matematika.

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i\left(x_j\right) = \sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i . x_j + b_i\right) = o_j,\ j \in [1, N] \tag{2.6}$$

dengan: w_i = $[w_{i1},\ w_{i2}, \dots, w_{i\tilde{N}}]^T$ adalah vektor bobot antara *input layer* dan *hidden node*

b_i = bias *hidden layer*

x_j = data ke-j

β_i = $[\beta_1,\ \beta_2, \dots, \beta_{\tilde{N}}]^T$

$w_i . x_j$ = *inner product* dari $w_i\ dan\ x_j$

o_j = *output* ke-j

SLFN standard dengan $\tilde{N}$ *hidden nodes* dan fungsi aktivasi $g(x)$ dapat mendekati N sampel dengan error rata – rata nol, $\sum_{i=1}^{\tilde{N}} \| o_j - t_j \| = 0$ terdapat β_i, w_i dan b_i sehingga

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i.x_j + b_i\right) = t_j,\ j \in [1, N] \tag{2.7}$$

dapat ditulis ,

$$H\beta_i = T$$

dengan

$$H = \begin{bmatrix} g\left(w_1.x_1 + b_1\right) & \cdots & g\left(w_{\tilde{N}}.x_1 + b_{\tilde{N}}\right) \\ \vdots & \ddots & \vdots \\ g\left(w_1.x_N + b_1\right) & \cdots & g\left(w_{\tilde{N}}.x_N + b_{\tilde{N}}\right) \end{bmatrix} \tag{2.8}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{\tilde{N}} \end{bmatrix} \text{ dan } T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$$

H adalah *output* matriks *hidden layer*, kolom ke-i dari H adalah *output hidden node* ke-i.

Algoritma ELM merupakan algoritma yang diperoleh dari solusi *minimum norm least square* SLFNs. Konsep utama dari ELM seperti yang disajikan dalam *paper* Huang (2006) adalah sebagai berikut:

Diberikan *training* set $\aleph = \{(x_j, t_j) | x_j \in \boldsymbol{R^{n \times m}}, t_j \in \boldsymbol{R^n},\ j \in [1, N]\}$, fungsi aktivasi $g(x)$, dan bilangan *hidden node* $\tilde{N}$

Step 1: masukkan secara random bobot w_i dan bias b_i, $i \in \left[1, \tilde{N}\right]$

Step 2: hitung *output* matriks *hidden layer H*

Step 3: hitung bobot *output* β

$$\beta = H^{\dagger}T \tag{2.9}$$

dengan $T = [t_1, \ldots\ldots, t_N]^T$,

$H^{\dagger}$ adalah *Generalized Inverse*

$H^{\dagger} = (H^T H)^{-1} H^T$ atau,

$H^{\dagger} = H^T (HH^T)^{-1}$

Pada ELM biasa, seluruh data langsung masuk dalam proses *training* secara bersama, akan tetapi dalam penelitian ini proses *training* dilakukan secara *sequential*, artinya data masuk proses *training* per kelompok data. Menurut

(Liang, 2006) proses *training* yang dilakukan secara *sequential* dapat mempengaruhi dari proses update bobot. Bobot *output* pada Persamaan 2.9 adalah

$$\beta = H^{\dagger}T$$

dengan

$$H^{\dagger} = (H^T H)^{-1} H^T$$

sehingga

$$\beta = (H^T H)^{(-1)} H^T T \qquad (2.10).$$

Persamaan 2.10 dinamakan *least square solution* dari $H\beta = T$. Implementasi sequential dari *least square solution* digunakan juga pada sequential *Extreme Learning Machine*.

Diberikan inisialisasi himpunan kelompok data $\aleph_0 = \{(x_j, t_j)\}_{j=1}^{N_0}$, seperti pada ELM biasa, ditentukan bobot *w* dan *b* secara random,

$$w = [x_{i1}, x_{i2}, \dots, x_{im}]^T$$
$$b = [t_1, t_2, \dots, t_{\tilde{N}}]^T.$$

selanjutnya dihitung matriks *hidden layer* H_0

$$H_0 = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_{N_0} + b_1) & \cdots & g(w_{\tilde{N}}.x_{N_0} + b_{\tilde{N}}) \end{bmatrix}$$

kemudian dicari vektor bobot *output*

$$\beta_0 = H_0^{\dagger} T_0$$
$$\beta_0 = K_0^{-1} H_0^T T_0$$

dengan

$$K_0 = H_0^T H_0.$$

Diberikan lagi himpunan kelompok data $\aleph_1 = \{(x_j, t_j)\}_{j=N_0+1}^{N_1+N_1}$, dengan bobot *w* dan *b* yang sama dan dengan cara yang sama dihitung matriks *hidden layer* H_1

$$H_1 = \begin{bmatrix} g\left(w_1.x_{N_0+1} + b_1\right) & \cdots & g\left(w_{\tilde{N}}.x_{N_0+1} + b_{\tilde{N}}\right) \\ \vdots & \ddots & \vdots \\ g\left(w_1.x_{N_1} + b_1\right) & \cdots & g\left(w_{\tilde{N}}.x_{N_1} + b_{\tilde{N}}\right) \end{bmatrix}$$

kemudian dicari vektor β

$$\beta_1 = K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \tag{2.11}$$

dengan

$$K_1 = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \tag{2.12}$$

Karena H_0 dan H_1 adalah merupakan matriks di dalam vektor, maka dari Persamaan 2.12 diperoleh

$$\begin{aligned} K_1 &= \begin{bmatrix} H_0^T & H_1^T \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \\ &= H_0^T H_0 + H_1^T H_1 \\ &= K_0 + H_1^T H_1 \end{aligned} \tag{2.13}$$

dari persamaan di atas dapat ditulis

$$K_0 = K_1 - H_1^T H_1 \tag{2.14}$$

selanjutnya

$$\begin{aligned} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} &= H_0^T T_0 + H_1^T T_1 \\ &= K_0 H_0^{-1} H_0^T T_0 + H_1^T T_1 \\ &= K_0 \beta_0 + H_1^T T_1 \\ &= \left(K_1 - H_1^T H_1\right) \beta_0 + H_1^T T_1 \\ &= K_1 \beta_0 - H_1^T H_1 \beta_0 + H_1^T T_1. \end{aligned} \tag{2.15}$$

Dari Persamaan 2.11 dan 2.14 diperoleh

$$\beta_1 = K_1^{-1}\begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix}^T$$

$$= K_1^{-1}\left(K_1\beta_0 - H_1^T H_1\beta_0 + H_1^T T_1\right)$$

$$= \beta_0 - K_1^{-1}H_1^T H_1\beta_0 + K_1^{-1}H_1^T T_1$$

$$= \beta_0 + K_1^{-1}H_1^T\left(T_1 - H_1\beta_0\right).$$

Dengan cara yang sama diberikan data ke $n+1$, sehingga diperoleh

$$K_{n+1} = K_n + H_{n+1}^T H_{n+1} \tag{2.16}$$

$$\beta_{n+1} = \beta_n + K_{n+1}^{-1}H_{n+1}^T\left(T_{n+1} - H_{n+1}\beta_n\right). \tag{2.17}$$

Karena pada Persamaan 2.17 yang digunakan adalah K_{n+1}^{-1} maka perlu dicari invers untuk K_{n+1}.

$$\left(K_{n+1}\right)^{-1} = \left(K_n + H_{n+1}^T H_{n+1}\right)^{-1}.$$

Berdasarkan *Sherman Morrison Woodbury Formula* (Golup,) maka diperoleh

$$K_{n+1}^{-1} = K_n^{-1} - K_n^{-1}H_{(n+1)}^T(I + H_{(n+1)}K_n^{-1}H_{(n+1)}^T)^{-1}H_{(n+1)}K_n^{-1}$$

dengan memisalkan $K^{-1} = P$, maka diperoleh

$$P_{n+1} = P_n - P_n H_{n+1}^T(I + H_{n+1}P_n H_{n+1}^T)^{-1}H_{n+1}P_n \tag{2.18}$$

$$\beta_{n+1} = \beta_n + K_{n+1}^{-1}H_{n+1}^T\left(T_{n+1} - H_{n+1}\beta_n\right) \tag{2.19}$$

2.11 *Imbalanced Data*

Imbalanced data atau data yang tak berimbang merupakan suatu kondisi dimana pada sebuah subset yang terdapat dalam suatu dataset memiliki jumlah *instance* yang jauh lebih kecil bila dibandingkan dengan subset lainnya (Melisa, 2013). *Imbalanced* data merupakan masalah yang sering muncul pada kasus klasifikasi, hal ini menyebabkan *classifier* cenderung memprediksi kelas mayoritas dan mengabaikan kelas minoritas sehingga keakuratan kelas minoritas sangat kecil (Trapsilasiwi, 2013).

Imbalanced data merupakan salah satu masalah yang sangat penting dalam masalah klasifikasi dan menjadi tantangan baru dalam *machine learning* (He, 2009). Masalah distribusi kelas data *imbalanced* dapat dilihat pada beberapa

masalah dunia nyata, termasuk dalam masalah kesehatan, keamanan, internet, keuangan dsb. Ketika mengklasifikasikan data dengan distribusi kelas yang *imbalanced* , algoritma pembelajaran biasanya memiliki kecenderungan condong ke kelas mayoritas dibandingkan kelas minoritas. Untuk memecahkan masalah data *imbalanced* ini ada beberapa cara, diantaranya adalah *Undersampling*, *oversampling*, dan modifikasi algoritma.

Perbandingan dan persentase suatu data dikatakan *imbalanced* ada beberapa macam. Suata data dikatakan *strong imbalanced* jika mempunyai perbandingan data positif dan negatifnya adalah 1:100, 1:1000, dan seterusnya (He,2009). Sedangakan Hulse (2007) menyebutkan bahwa suatu data dikatakan *imbalanced* jika hanya terdapat 35% dari data yang merupakan data positif. Sedangkan menurut Batuwitage (2010), data *imbalanced* jika 40% persen dari data keseluruhan adalah data positif dan sisanya adalah data negatif maka data termasuk *imbalanced* data.

2.11.1 *Performance Measure* (Ukuran Performa)

Imbalanced data merupakan kasus khusus dalam *Machine Learning*. Ukuran evaluasi memainkan peran penting dalam *machine learning*. Ukuran tersebut digunakan untuk mengevaluasi dan mengarahkan algoritma pembelajaran. Jika pilihan ukuran mengabaikan kelas minoritas, maka algoritma pembelajaran tidak akan mampu menangani masalah *imbalanced data* dengan baik. Ukuran yang umum digunakan untuk dalam penelitian biasanya adalah tingkat klasifikasi keseluruhan yaitu akurasi. Namun pada *imbalanced* dataset, tingkat klasifikasi keseluruhan tidak lagi menjadi ukuran yang cocok, karena kelas minoritas tidak berpengaruh pada akurasi dibandingkan dengan kelas mayoritas.

Oleh karena itu, ukuran lainnya telah dikembangkan untuk menilai kinerja *classifier* untuk data yang *imbalanced*. Berbagai ukuran yang umum didefinisikan berdasarkan *confusion matrix*. *Confusion matrix* untuk klasifikasi biner ditunjukkan dalam Tabel 2.2.

Tabel 2.2 *Confusion matrix* untuk klasifikasi biner

		Nilai Sebenarnya	
		True	*False*
Prediksi	*True*	TP *(True Positive)*	FP *(False Positive)*
	False	FN *(False Negative)*	TN *(True Negative)*

Diantara berbagai kriteria evaluasi, ukuran yang paling relevan dengan data yang *imbalanced* yaitu *precision*, *recall*, *sensitifity*, *specificity*, dan *geometric mean* (*G-mean*).

a. *Precision*

Precision adalah presentase dari data yang diprediksi benar oleh *classifier* yang bernilai benar. Rumus dari *precision* adalah:

$$precision = \frac{TP}{TP + FP}. \tag{2.10}$$

b. *Recall*

Recall adalah porsi dari data sampel yang diprediksi benar oleh *classifier*. Rumus dan perhitungan *recall* adalah sebagai berikut:

$$recall = \frac{TP}{TP + FN}. \tag{2.11}$$

c. *Geometric mean* (*G-mean*)

Ukuran ini digunakan ketika performa dari kedua kelas yang bersangkutan dan diharapkan tinggi secara bersamaan. *Geometric mean* (Kubat dan Matwin, 1997) dan telah digunakan beberapa peneliti untuk mengevaluasi *classifier* pada dataset yang *imbalanced* . *G-mean* mengindikasikan keseimbangan antara kinerja klasifikasi pada kelas mayoritas dan minoritas. Ukuran *G-mean* diambil berdasarkan s*ensitifity* (akurasi dari data positif) dan s*pecificity* (akurasi data negatif).

sensitifity=recall

$$specificity = 1 - \frac{TP}{FP + TN} \tag{2.12}$$

$$G - mean = \sqrt{sensitivity \times specificity} \tag{2.13}$$

(He, 2009).

2.11.2 Metode *Sampling*

Pendekatan umum untuk menangani masalah *imbalanced data* adalah penanganan sampel. Ide utamanya adalah untuk *preprocessing* pelatihan diatur untuk meminimalkan perbandingan antara kelas. Dengan kata lain, metode pengambilan sampel mengubah jumlah awal distribusi minoritas dan mayoritas kelas dalam pelatihan yang diatur untuk mendapatkan data seimbang dari contoh di masing-masing kelas. Beberapa metode *sampling* untuk menangani masalah *imbalanced data*, diantaranya adalah *Undersampling*, *oversampling*, dan beberapa metode *sampling* yang lain.

a. *Undesampling*

Undersampling adalah metode non-*heuristik* yang menghilangkan sampel dari kelas mayoritas untuk menyeimbangkan distribusi kelas. Logika di balik ini adalah mencoba untuk menyeimbangkan kumpulan data untuk mengatasi masalah dari algoritma (Bekkar, 2013). Japkowicz (2000) membedakan dua jenis *Undersampling*. *Random Undersampling* (RUS), yang mengambil atau membuang data sampel secara acak dari kelas mayoritas. *Fokus Undersampling* (FUS), yang tidak melibatkan sampel kelas mayoritas yang berada di perbatasan antar kelas.

Undersampling adalah sebuah metode yang efisien untuk learning kelas *imbalanced* . Metode ini menggunakan *subset* dari kelas mayoritas untuk melatih suatu *classifier* (Ganganwar, 2012). Dengan kata lain konsep *Undersampling* adalah mengurangi data mayoritas untuk data menjadikan data menjadi seimbang.

Undersampling cocok untuk diaplikasikan pada data skala besar, dimana jumlah sampel mayoritas sangat besar sehingga mengurangi waktu pelatihan dan penyimpanan. Namun, kelemahan *Undersampling* adalah dapat menyebabkan hilangnya data yang kelas informatif pada kelas mayoritas sehinggga menurunkan performa *classifier* (Nguyen, 2009).

b. *Oversampling*

Oversampling adalah pendekatan yang meningkatkan proporsi dari kelas minoritas dengan menduplikasi pengamatan terhadap kelas tersebut. Sebagaimana *Undersampling, oversampling* juga dibedakan menjadi dua macam. *Random oversampling* didasarkan pada pilihan secara acak pada proses duplikasi. *Focus*

Oversampling menduplikasi pengamatan di perbatasan antara kelas mayoritas dan minoritas (Bekkar, 2013).

Oversampling adalah meningkatkan atau menambah jumlah data dari data minoritas dengan mereplikasi data sampel tersebut (Japkowicz dan Stephen, 2002). Keuntungan dari *oversampling* adalah tidak ada informasi yang hilang, semua sampel masuk dalam proses *training* . Namun, *oversampling* juga memiliki kelemahan sendiri, dengan menciptakan contoh pelatihan tambahan, *oversampling* menyebabkan biaya komputasi yang lebih tinggi. Jika, beberapa sampel kelas minoritas mengandung kesalahan pelabelan, hal tersebut dapat memperburuk performa klasifikasi pada kelas minoritas (Chawla, 2010).

2.12 *Cross Validation*

Cross Validation merupakan salah satu teknik untuk memvalidasi keakuratan sebuah model yang dibangun berdasarkan data set tertentu. Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang boleh jadi belum pernah muncul di dalam data set. Data yang digunakan dalam proses pembangunan model disebut data latih atau *training* , sedangkan data yang digunakan untuk memvalidasi model disebut sebagai data *test* (Dian, 2015).

Dalam metode *cross validation*, ada beberapa macam *cross validation* diantaranya adalah metode *Two-Fold cross validation*, metode *k-fold cross-validation*, dan metode *leave-One-Out Cross Validation*.

a. Metode *Two-Fold cross validation*

Two-Fold cross validation adalah variasi sederhana *K-fold cross validation*. Metode ini disebut juga *holdout method*. Untuk setiap *fold*, secara acak ditetapkan titik data untuk dua kelompok data $d0$ dan $d1$, sehingga kedua kelompok data mempunyai ukuran yang sama. Kemudian data pada $d0$ diguanakan untuk *training* dan data $d1$ untuk *testing*, Selanjutnya diikuti oleh *training* data $d1$ dan *testing* data $d0$. Hal ini memiliki keuntungan data *training* dan *testing* terdiri dari dua data yang besar (Syed, 2011).

b. Metode *k-fold cross-validation*

Menurut Fu (1994), *K-fold cross validation* mengulang *k* kali untuk membagi sebuah himpunan contoh secara acak menjadi *k* subset yang saling bebas, setiap ulangan disisakan satu subset untuk pengujian dan subset lainnya untuk pelatihan. Keuntungan dari metode ini lebih berulang secara acak setap subset, artinya bahwa semua data digunakan untuk pelatihan dan validasi, dan masing-masing data digunakan untuk validasi tepat sekali.

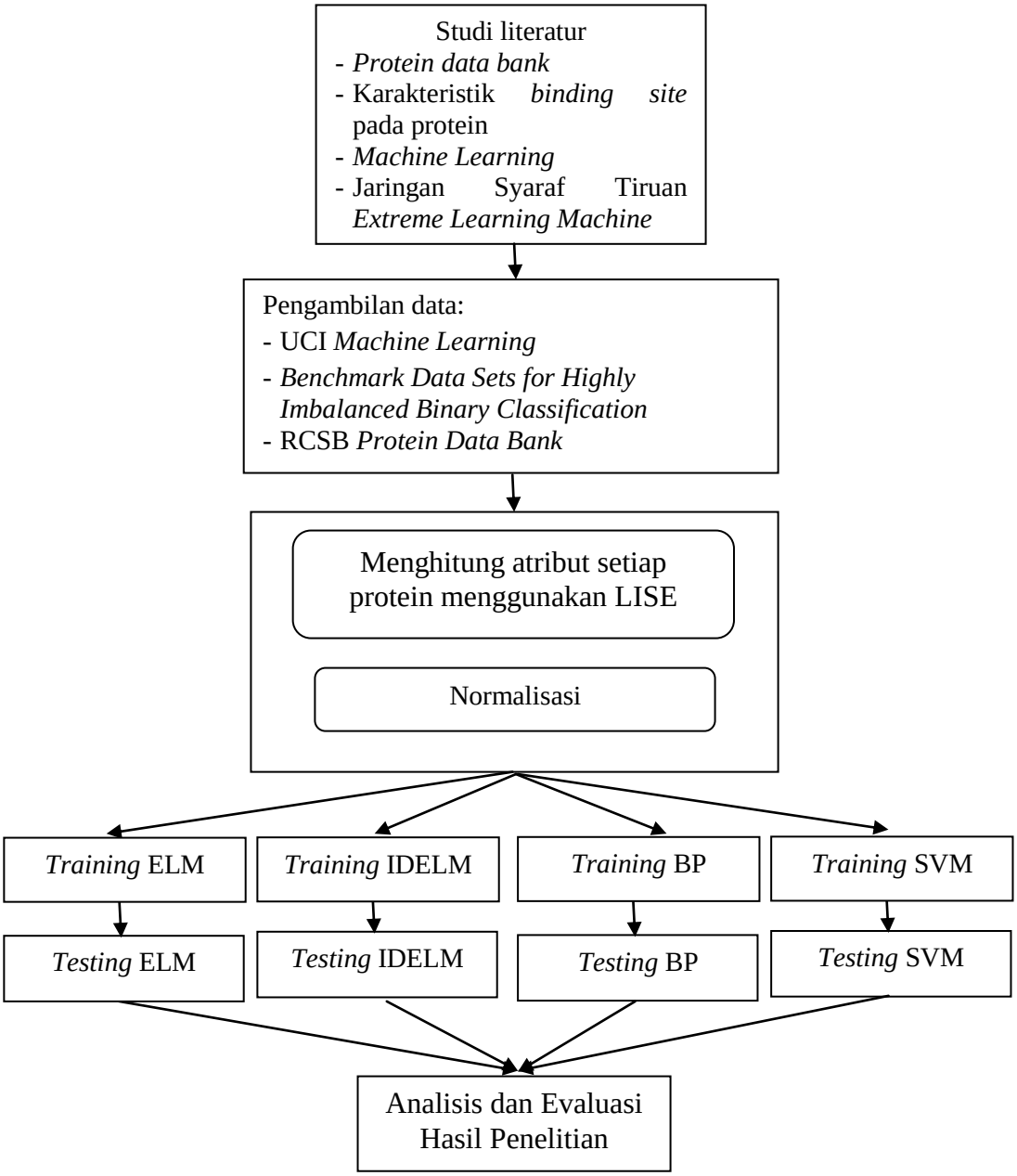
c. Metode *leave-One-Out Cross Validation*

Leave-One –Out adalah bentuk khusus dari cross-validation, yaitu jumlah *fold* sama dengan jumlah data *training* . Artinya, dari seluruh data yang ada, tinggalkan satu data untuk validasi dan data lainnya untuk membuat model. Satu data yang tersisa akan diprediksi dari model yang diperoleh dan dilihat sejauh mana keakuratannya dengan data yang sebenarnya. Metode ini dilakukan sampai seluruh data tervalidasi. Total selisih data prediksi dan data sebenarnya (total residuals) akan digunakan sebagai kriteria kevalidan model tersebut. Oleh karena itu, metode ini membutuhkan biaya komputasi yang sangat mahal. Sehingga, metode ini diutamakan untuk data yang ukurannya kecil (Syed, 2011).

BAB 3
METODE PENELITIAN

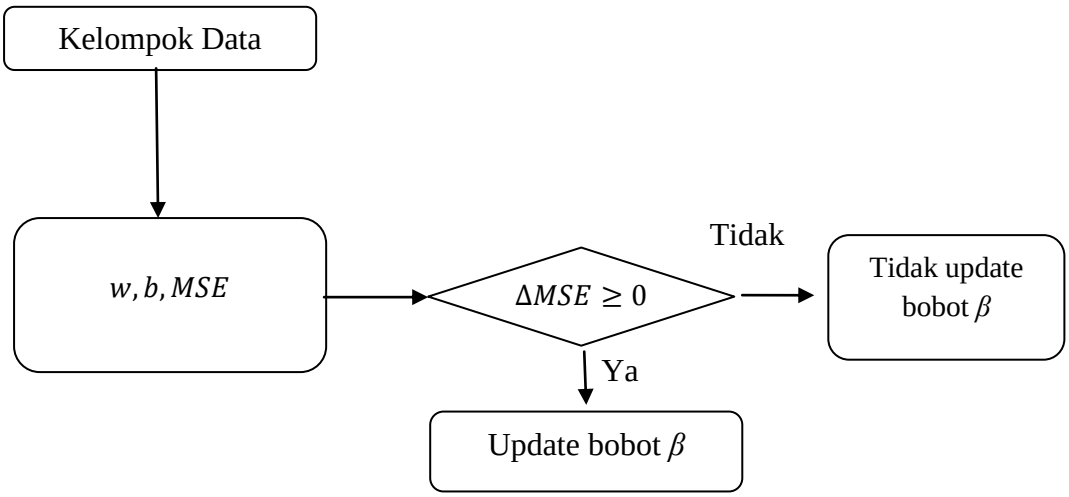
3.1 Tahapan Penelitian

Secara umum, tahapan yang dilakukan dalam penelitian ini disampaikan pada diagram alir penelitian seperti pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Diagram Alir Penelitian

Catatan: BP = *Backpropagation*
SVM = *Support Vector Machine*



Gambar 3.2 Proses *Training* ELM terintegrasi

Keterangan:

$$\Delta MSE = MSE_{t+1} - MSE_t \quad (3.1)$$

Berdasarkan diagram alir pada Gambar 3.1 dan Gambar 3.2, maka dijelaskan lebih rinci sebagai berikut.

3.1.1 Studi literatur

Pada tahapan ini dilakukan observasi permasalahan, identifikasi permasalahan serta pemahaman teori tentang pencarian *binding site* protein-ligan dan algoritma penyelesaiannya. Selanjutnya mencari referensi yang menunjang penelitian. Referensi yang digunakan adalah paper, jurnal, dan buku–buku yang menunjang penelitian.

3.1.2 Pengambilan data

Dalam penelitian ini tidak hanya melakukan prediksi *binding site* protein-ligan dengan menggunakan ELM, tetapi juga melakukan Integrasi seleksi data dan ELM. Karena melakukan Integrasi seleksi data dan ELM, maka sebelum program digunakan untuk prediksi *binding site* protein-ligan terlebih dahulu diuji dengan menggunakan data dengan karakter yang mirip dengan data *binding site* protein-ligan yaitu data yang *imbalanced*. Sehingga data yang digunakan dalam penelitian ini adalah

a. Data untuk menguji Integrasi seleksi data dan klasifikasi pada ELM dari UCI *Machine Learning* dan *Benchmark Data Sets for Highly Imbalanced Binary Classification.*
b. Data eksperimental, yaitu data protein yang didapat dari webserver RCSB Protein Data Bank.

3.1.3 Pembagian Data

Proses *training* dan *testing* diperlukan pada proses pengklasifikasian dengan *Extreme Learning Machine*. Proses *training* untuk mendapatkan pola dari data dan kelas, sedangkan *testing* digunakan untuk mengevaluasi kemampuan ELM. Dalam penelitian ini pemilihan data *training* dan *testing* adalah sebagai berikut:

a. Pada uji dengan menggunakan *benchmark data*, untuk membagi data *training* dan data *testing* digunakan 5-*fold cross validation* artinya 80% data untuk *training* dan 20% *testing*.
b. Untuk prediksi *binding site* protein-ligan karena dalam penelitan ini ada 17 data protein yang digunakan. Karena data diambil dari beberapa jenis protein, maka protein dikelompokkan terlebih dahulu berdasar jenisnya, selanjutnya dilakukan *training* dan *testing* per jenis protein. Sehingga untuk pembagian data *training* dan *testing* digunakan 1 protein untuk *testing* dan 1 protein untuk *training* yang diambil dari protein dalam satu jenis. Karena 1 protein *training* dan 1 protein *testing*, maka proses *training* dilakukan sebanyak sisa data protein dalan satu jenis protein. *Testing* dalam hal ini adalah untuk prediksi *binding site* protein-ligan dari protein yang digunakan sebagai *testing*.

3.1.4 *Prepocessing*

Preprocessing ini bertujuan untuk menyiapkan data sebelum sebelum masuk proses *training*, Langkah-langkah *preprocessing* adalah sebagai berikut.

a. Menghitung atribut setiap titik grid

Dalam hal ini langkah pembuatan 3D grid dan perhitungan atribut pada setiap titik grid menggunakan program LISE. Dari sebuah file PDB data protein disubmit pada LISE selanjutnya oleh LISE dianalisis sehingga mendapatkan *output* berupa analisis prediksi *binding site* berdasarkan analisis geometri dan analisis *sequence*.

b. Normalisasi Data

Data yang digunakan pada *Extreme Learning Machine* sebaiknya dinormalisasi sehingga mempunyai nilai dengan *range* tertentu. Dalam penelitian ini digunakan rumus normalisasi minmax (Zhu, 2013) sbb:

$$x_{normalized} = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \times \left(\max_{new} - \min_{new} \right) \right) + \min_{new} \qquad (3.2)$$

Keterangan:

$x_{normalized}$ = data ternormalisasi

x = data yang akan dinormalisasi

x_{min} = nilai minimum dari data

x_{min} = nilai maksimum dari data

max_{new} = nilai maksimal baru yang diinginkan

min_{new} = nilai minimal baru yang diinginkan

3.1.5 Pelatihan (*training*) Integrasi Seleksi Data dan *Extreme Learning Machine*

Konsep pembelajaran yang digunakan pada *Extreme Learning Machine* meminimalkan *cost function* dengan menggunakan *minimum least square*. Tujuan dari proses ini adalah untuk mendapatkan bobot *output* dengan tingkat kesalahan yang rendah.

Dalam penelitian ini *undersampling* digunakan untuk mengatasi masalah data *imbalance*. Selanjutnya, proses *undersampling* diintegrasikan dengan proses *training*. Pengelompokan data dilakukan secara *sequential* sebagaimana dilakukan oleh Liang (2006) yaitu konsep *online sequential*

ELM dengan *chunk-by-chunk* atau biasa disebut dengan *block of data*. Dalam penentuan banyak data setiap kelompok data disyaratkan harus kurang dari banyak data pada kelas minoritas. Kelas minoritas merupakan kelas yang banyak datanya jauh lebih sedikit dibandingkan kelas yang lain. Pada proses *training* jumlah *hidden node* dan fungsi aktivasi dari IDELM harus ditentukan terlebih dahulu, dalam hal ini fungsi aktivasi yang digunakan adalah fungsi sigmoid (Siang, 2009).

$$g(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

Bobot w_i dan b_i ditentukan secara random, sedangkan bobot β merupakan perkalian *Moore–Penrose generalized inverse* dari H dan target. Secara matematis (Huang, 2006) dapat ditulis sebagai berikut :

$$\beta = H^{\dagger}T \tag{3.4}$$

dengan:

$$H = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \cdots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}$$

$g(w_1.x_1 + b_1)$ adalah fungsi sigmoid,

$T = [t_1, \dots\dots, t_N]^T$

Moore–Penrose generalized inverse dari H dirumuskan sebagai berikut (Ferrari, 2005):

$H^{\dagger} = (H^T H)^{-1} H^T$

atau

$H^{\dagger} = H^T (HH^T)^{-1}$

Untuk menghitung *output* dari ELM dapat digunakan rumus sebagai berikut:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i.x_j + b_i) = y_j$$

atau

$$Y = H.\beta \tag{3.5}$$

Keterangan:

Y = *output*

H = matriks *hidden layer*

β = bobot *output*

Selanjutnya dihitung *Mean Square Error* (MSE) (Makridakis, 1999).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - t_i\right)^2 \tag{3.6}$$

MSE dalam penelitian ini digunakan sebagai referensi apakah kelompok data diplih sebagai *prototype* atau data *training* . Dalam penelitian ini, jika MSE kelompok data ke *k* lebih kecil dibandingkan MSE kelompok data ke *k+1* maka kemompok data ke *k+1* digunakan sebagi data *training* . Dengan kata lain jika $\Delta MSE \geq 0$ maka kelompok data dipilih sebagai data *training*. Sedangkan, jika $\Delta MSE < 0$ maka kelompok data tersebut dihapus atau tidak digunakan sebagai data *training*. Selanjutnya, jika $\Delta MSE \geq 0$, maka dilakukan update bobot.

$$\beta_{n+1} = \beta_n + K_{n+1}^{-1} H_{n+1}^T \left(T_{n+1} - H_{n+1}\beta_n\right) \tag{3.7}$$

dengan

$$P_{(n+1)} = P_n - P_n H_{(n+1)}^T (I + H_{(n+1)} P_n H_{(n+1)}^T)^{-1} H_{(n+1)} P_n \tag{3.8}$$

dan sebagai inisialisasi *P*

$$P_0 = \left(H_0{}^T H_0\right)^{-1}. \tag{3.9}$$

Keterangan:

$\beta^{(k+1)}$ = bobot β untuk data ke *k+1*

$\beta^{(k)}$ = bobot β untuk data ke *k*

H = matriks *hidden layer*

P_0 = inisalisasi P

3.1.6 *Testing* Integrasi Seleksi Data dan *Extreme Learning Machine*

Berdasarkan bobot yang didapatkan dari proses *training*, maka tahap selanjutnya adalah melakukan *testing* dengan IDELM. *Testing* ini dilakukan bertujuan untuk menguji seberapa baik kemampuan integrasi seleksi data dan ELM (IDELM).

3.1.7 Menghitung Akurasi

Untuk membandingkan dan mengevaluasi IDELM dengan metode yang lainnya digunakan perhitungan performa, dalam hal ini adalah akurasinya. Dalam penelitian ini diaplikasikan dua ukuran dalam metode dan dataset yang berbeda.

Pada dataset (Wong, 2013), *grid point* digunakan untuk merepresentasikan *potensial binding site*. Jika sebuah *grid point* dicluster ternyata tidak cocok untuk *binding site*, akan diberikan nilai 0. Sehingga prediksi dari *binding site* ligan direpresentasikan oleh bilangan *grid point* selain nol, dalam hal ini diberikan nilai 1.

Untuk mengukur performa dari klasifikasi data pengujian *imbalance* dalam hal ini digunakan *confusion matrix*, *precision* dan *recall*, *specificity*, dan *G-mean*. Tabel *confusion matrix* Dapat dilihat pada Tabel 3.1, sedangkan untuk pengukuran performa seperti pada Persamaan 3.10 sampai Persamaan 3.14.

Tabel 3.1 *Confusion Matrix*

		Nilai Sebenarnya	
		True	*False*
Prediksi	*True*	TP *(True Positive)*	FP *(False Positive)*
	False	FN *(False Negative)*	TN *(True Negative)*

$$\text{Akurasi} = \frac{TP + TN}{\text{total sampel}} \tag{3.10}$$

dengan: TP = nilai true positive

FP = false positive

FN =false negative

Precision adalah porsi dari prediksi data positif yang dikenali benar oleh algoritma. *Precision* dari data *testing* di atas dirumuskan sebagai berikut:

$$precision = \frac{TP}{TP + FP} \tag{3.11}$$

Recall adalah porsi dari data sampel yang diprediksi benar oleh algoritma. Rumus dan perhitungan *recall* adalah sebagai berikut:

$$recall = \frac{TP}{TP + FN} \tag{3.12}$$

G-mean (*Geometric mean*) merupakan salah satu alat ukur performa untuk evaluasi data yang mempunyai karakteristik *imbalance*.

$$G - mean = \sqrt{sensitivity \times specificity} \tag{3.13}$$

Untuk menghitung *G-mean* terlebih dahulu *sensitifity* dan *specificity*. *Sensifity* adalah untuk mengukur akurasi sampel positif, untuk menghitung *sensitifity* sama dengan perhitungan *recall*. Sedangkan *specificity*, adalah akurasi untuk sampel negatif, rumus untuk menghitung *specificity* dan perhitungan untuk data *testing* adalah sebagai berikut:

$$specificity = 1 - \frac{TP}{FP + TN} \tag{3.14}$$

BAB 4
HASIL DAN PEMBAHASAN

Dalam penelitian ini terdapat dua bagian penelitian, yaitu Integrasi seleksi data dan *Extreme Learning Machine* dan pediksi *binding site* protein-ligan dengan menggunakan integrasi seleksi data dan *Extreme Learning Machine*.

4.1 Pengumpulan Data

Dataset yang digunakan merupakan data eksperimental, yaitu data protein murni yang didapat dari *webserver* RCSB *Protein Data Bank*. Selanjutnya untuk menghitung atribut protein tersebut digunakan *webserver* LISE, dengan cara menginputkan file PDB yang diperoleh pada *webserver* LISE untuk dianalisis. Penelitian ini mengerjakan dua hal, yaitu perbaikan ELM dan mengaplikasikan perbaikan ELM untuk prediksi *binding site* protein-ligan.

Sebelum integrasi seleksi data dan klasifikasi pada ELM (IDELM) diaplikasikan untuk prediksi *binding site* protein-ligan terlebih dahulu perbaikan ELM diuji dengan menggunakan *benchmark data*. Pengujian tersebut bertujuan untuk melihat performa dari integrasi seleksi data dan ELM. *Benchmark data* diambil dari UCI *Machine Learning* dan *Benchmark Data Sets for Highly Imbalanced Binary Classification*. Detail data yang digunakan untuk ujicoba IDELM yang dilakukan dapat dilihat pada Tabel 4.1.

Tabel 4.1 *Benchmark Data* untuk ujicoba IDELM

Data	Ukuran Data	Banyak Atribut	Banyak Data -	Banyak Data +
Wilt (W)	4339	5	356	699
Spambase(Sp)	4601	51	1813	2788
Qsar	1055	41	70	144
Balance(B)	626	4	76	138
Glass1(G1)	214	9	17	197
Glass2(G2)	214	9	74	4265
Glass3(G3)	214	9	26	600
Yeast(Y)	1485	8	210	3968
Abalone(Y)	4175	8	33	1452

Tabel 4.1 *Benchmark Data* untuk ujicoba IDELM (lanjutan)

Data	Ukuran Data	Banyak Atribut	Banyak Data +	Banyak Data -
Solar Flare(SF)	1390	10	44	1346
Mammography(M)	11183	6	260	10923
Forest Cover(FC)	2267	12	189	2078
Letter Img(LI)	20000	16	734	19266

Setelah diuji akurasi perbaikan ELM dengan menggunakan data pada Tabel 4.1, selanjutnya integrasi seleksi data dan ELM digunakan untuk prediksi *binding site* protein-ligan. Detail data yang digunakan untuk menguji akurasi prediksi *binding site* protein-ligan dapat dilihat pada tabel 4.2.

Tabel 4.2 Besar data dan jenis data protein

No	Jenis Protein	Protein ID	Ukuran Data	Banyak Data +	Banyak Data -
1	Hydrolase	4TPI	3042	776	2266
2		2ZAL	5286	688	4598
3		2V8L	2060	1030	1030
4		1WYW	9616	827	8789
5		1RN8	2233	918	1315
6		1C1P	4797	704	4093
7		1YBU	5909	718	5191
8	Oxidoreductase	3D4P	6204	988	5216
9		1A4U	4663	737	3926
10		2WLA	2444	846	1598
11	Transferase	2GGA	4146	316	3830
12		1SQF	4365	934	3431
13		1O26	9272	1332	7940
14		1G6C	4504	724	3780
15		1BJ4	4205	477	3728
16	Ligase	1U7Z	6144	731	5413
17		1ADE	9072	805	8267

Data protein dikelompokkan menjadi beberapa kelompok, karena penelitian ini terkait juga dengan masalah kimia dan biologi, sehingga perlu dipisahkan antar jenis protein. Pemisahan berdasarkan jenis protein tersebut dilakukan, karena karakter setiap protein berbeda pada setiap jenis protein, terutama masalah energi interaksi dan *sequence*.

4.2 Integrasi Seleksi Data dan Klasifikasi pada *Extreme Learning Macine* (ELM)

Extreme Learning Machine (ELM) adalah bagian dari algoritma Jaringan Syaraf Tiruan yang termasuk dalam metode terawasi. ELM merupakan algoritma klasifikasi yang proses learningnya sangat cepat dibanding dengan algoritma Jaringan Syaraf Tiruan yang lainnya, karena ELM tidak membutuhkan proses iterasi untuk menentukan nilai parameternya. Metode ELM termasuk metode Jaringan Syaraf Tiruan yang dapat dikatakan metode yang masih baru. Metode yang dikenalkan Huang (2004) ini merupakan metode yang didesain untuk data yang seimbang (*balanced data*) yaitu data yang perbandingan data positif dan negatif nya tidak terlalu banyak. Dalam kasus pencarian *binding site*, data yang digunakan termasuk data yang tidak seimbang ata *imbalanced data*, sehingga perlu diberikan perlakuan khusus, sehingga dilakukan perlakuan khusus pada ELM yang digunakan untuk *imbalanced* data.

Integrasi seleksi data dan ELM dalam penelitian ini bertujuan agar ELM bisa memiliki performa yang baik untuk data yang *imbalanced*. Sehingga dapat diaplikasikan untuk prediksi *binding site* protein ligan, karena karakteristik data yang digunakan untuk prediksi *binding site* adalah *imbalanced data*. Proses *training* adalah proses yang sangat penting dalam *machine learning*, karena pada proses *training* ini suatu mesin atau sistem diajarkan untuk mengenali pola yang diinginkan. Dalam hal ini adalah mengenali pola *binding site* atau bukan *binding site* dari data yang dimasukkan. Protein merupakan suatu senyawa yang sangat besar, sedangkan bagian yang berpotensi sebagai *binding site* merupakan bagian kecil dari suatu protein. Masalah tersebut merupakan kasus *imbalanced* , karena data yang berpotensi sebagai *binding site* dan yang bukan *binding site* perbandingannya jauh.

Solusi untuk mengatasi masalah data yang *imbalanced* adalah dengan seleksi data. Salah satunya dari jenis seleksi data yaitu dengan melakukan *Undersampling*. *Undersampling* merupakan suatu teknik seleksi data dengan mengurangi jumlah data mayoritas, sehingga perbandingan data mayoritas dan minoritas tidak terlalu jauh.

Dalam penelitian ini proses seleksi data terintegrasi dengan *Extreme Learning Machine,* dengan tujuan untuk menghindari masalah inkonsistensi ketika proses seleksi data dan proses *training*. Seleksi data dilakukan untuk mengatasi masalah *imbalanced* data. Seleksi data dilakukan secara kelompok, artinya ketika proses learning data dimasukkan secara *sequential* dengan syarat data yang masuk harus kurang dari banyak data minoritas. Untuk melakukan seleksi data digunakan *Mean Square Error*. Dengan bobot antara *input layer* dan *hidden layer* serta bias yang sama, menggunakan perlakuan yang sama juga, jika diperoleh $\Delta MSE = MSE_{n+1} - MSE_n \geq 0$, maka ada suatu karakter unik dalam data ke n+1 yang menyebabkan MSE naik, sehingga jika $\Delta MSE \geq 0$ maka data digunakan sebagai data *prototype* dan dilakukan update bobot.

Proses seleksi data tidak selalu bisa membuat data menjadi *balanced* secara ukuran, akan tetapi, dengan pembelajaran yang *sequential* dan adanya seleksi data dapat menurunkan tingkat *imbalanced* data yang sebelumnya dan mendapatkan data yang proporsinya baik, tanpa mengurangi atau menghilangkan informasi penting dalam data. Hal tersebut diadopsi dari konsep pada SVM dengan *support vector* nya.

4.2.1 Pembagian Data *Training* dan Data *Testing*

Proses *training* dan *testing* diperlukan pada proses pengklasifikasian dengan menggunakan integrasi seleksi data dan *Extreme Learning Machine*. Proses *training* untuk mendapatkan model, sedangkan *testing* digunakan untuk mengevaluasi kemampuan ELM. Oleh karena itu data dibagi menjadi data *training* dan data *testing*.

a. Pada uji dengan menggunakan *benchmark data*, untuk membagi data *training* dan data *testing* digunakan *k-fold cross validation*. Dalam penelitian ini digunakan 5-*fold cross validation* untuk pembagian

kelompok data *training* dan *testing* artinya adalah 80% data untuk *training* dan 20% *testing*. Prinsip dari 5-*fold cross validation* adalah, data keseluruhan dibagi menjadi lima kelompok data, 1 kelompok data untuk *training* dan empat kelompok data yang lain untuk *testing*. *Training* setiap dataset dilakukan sebanyak lima kali, sehingga setiap kelompok data pernah menjadi data *training* maupun data *testing*.

b. Dalam penelitan ini ada 17 data protein yang digunakan untuk prediksi *binding site* protein-ligan. Oleh karena itu, untuk pembagian data *training* dan *testing* digunakan 1 protein untuk *testing* dan 1 protein untuk *training*, protein untuk *training* diambil dari sisa protein yang lain yang yang sejenis. Sehingga proses *training* dilakukan sebanyak data protein sisa dari protein yang sejenis. *Testing* dalam hal ini adalah untuk prediksi *binding site* protein-ligan dari protein yang digunakan sebagai *testing*.

4.2.2 *Preprocessing*

Sebelum data protein diprediksi suatu *binding site* nya, maka data protein terlebih dahulu melalui beberapa langkah *preprocessing*.

a. Perhitungan atribut

Setelah data protein diperoleh dari dataset RCSB Protein Data Bank yang berupa file pdb, data protein tersebut dihitung atributnya. Perhitungan attribute dalam penelitian ini dengan menggunakan webserver LISE. Yaitu webserver yang digunakan untuk prediksi *binding site* dengan analisis geometri, *sequence*, dan energi. Konsep dasar dari webserver LISE ini adalah membuat titik grid 3D disekitar protein yang berada pada bagian protein yang cekung. Selanjutnya dari titik grid tersebut dianalisis berdasarkan interaksi antar atom, *sequence* dan geometri. Hasil analisis dari LISE diperoleh dua data, yaitu data jarak grid ke atom ligan dan skor grid. Contoh hasil analisis dari webserver LISE dapat dilihat pada Gambar 4.1.

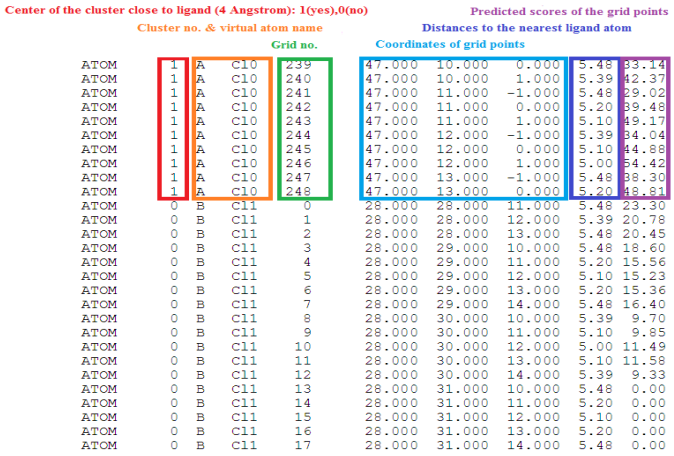
b. Normalisasi

Setelah diperoleh hasil atribut dari LISE, data atribut tersebut dinormalisasi. Normalisasi dalam penelitian ini bertujuan supaya data

mempunyai *range* data yang seragam. Untuk normalisasi data digunakan Persamaan 3.2. Dalam penelitian ini range yang normalisasi bertujuan untuk menseragamkan range data.Range yang diinginkan adalah antara 0 sampai 1, maka nilai $\max_{new} = 1$ sedangkan nilai $\min_{new} = 0$. Sehingga rumus normalisasi dapat ditulis:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

x adalah data yang akan dinormalisasi, x_{min} adalah nilai minimal semua data, x_{max} adalah nilai data maksimal dari semua data. Data yang dinormalisasi hanya data atribut yang telah diperoleh dari webserver LISE, yaitu jarak antara titik grid ke atom ligan terdekat dan skor grid.



ATOM	1	A	C10	239	47.000	10.000	0.000	5.48	33.14
ATOM	1	A	C10	240	47.000	10.000	1.000	5.39	42.37
ATOM	1	A	C10	241	47.000	11.000	-1.000	5.48	29.02
ATOM	1	A	C10	242	47.000	11.000	0.000	5.20	39.48
ATOM	1	A	C10	243	47.000	11.000	1.000	5.10	49.17
ATOM	1	A	C10	244	47.000	12.000	-1.000	5.39	34.04
ATOM	1	A	C10	245	47.000	12.000	0.000	5.10	44.88
ATOM	1	A	C10	246	47.000	12.000	1.000	5.00	54.42
ATOM	1	A	C10	247	47.000	13.000	-1.000	5.48	38.30
ATOM	1	A	C10	248	47.000	13.000	0.000	5.20	48.81
ATOM	0	B	C11	0	28.000	28.000	11.000	5.48	23.30
ATOM	0	B	C11	1	28.000	28.000	12.000	5.39	20.78
ATOM	0	B	C11	2	28.000	28.000	13.000	5.48	20.45
ATOM	0	B	C11	3	28.000	29.000	10.000	5.48	18.60
ATOM	0	B	C11	4	28.000	29.000	11.000	5.20	15.56
ATOM	0	B	C11	5	28.000	29.000	12.000	5.10	15.23
ATOM	0	B	C11	6	28.000	29.000	13.000	5.20	15.36
ATOM	0	B	C11	7	28.000	29.000	14.000	5.48	16.40
ATOM	0	B	C11	8	28.000	30.000	10.000	5.39	9.70
ATOM	0	B	C11	9	28.000	30.000	11.000	5.10	9.85
ATOM	0	B	C11	10	28.000	30.000	12.000	5.00	11.49
ATOM	0	B	C11	11	28.000	30.000	13.000	5.10	11.58
ATOM	0	B	C11	12	28.000	30.000	14.000	5.39	9.33
ATOM	0	B	C11	13	28.000	31.000	10.000	5.48	0.00
ATOM	0	B	C11	14	28.000	31.000	11.000	5.20	0.00
ATOM	0	B	C11	15	28.000	31.000	12.000	5.10	0.00
ATOM	0	B	C11	16	28.000	31.000	13.000	5.20	0.00
ATOM	0	B	C11	17	28.000	31.000	14.000	5.48	0.00

Gambar 4.1 Hasil analisis webserver LISE (Sumber: webserver LISE)

4.2.3 *Training* Integrasi Seleksi Data dan *Extreme Learning Machine*

Sebelum data masuk proses *training* dataset terlebih dahulu dibuat menjadi berapa kelompok. Pembagian kelompok data didasarkan pada konsep *sequential learning* untuk *feedforward network* seperti yang dilakukan oleh Nan-Ying Liang (2006), dengan menggunakan prinsip *chunk-by-chunk* atau biasa disebut dengan *block of data*. Data diambil secara terurut seperti pada *Sequential Extreme Learning Machine*. Dalam penentuan banyak data, pada IDELM setiap kelompok data disyaratkan harus kurang dari banyak data pada kelas minoritas. Kelas minoritas merupakan kelas yang banyak datanya jauh lebih sedikit dibandingkan kelas yang lain. Karena pada *benchmark data* ada beberapa data yang memiliki

banyak data minoritas di bawah 20 data, maka pada penelitian ini, untuk *benchmark data* digunakan *sequential data* 10-*by*-10. Sedangkan, pada data protein digunakan *sequential data* 100-*by*-100, hal tersebut dikarenakan data *binding site* protein-ligan memiliki data minoritas rata-rata di atas 500 data.

Data yang sudah dinormalisasi dan dibagi menjadi beberapa kelompok selanjutnya masuk pada proses *training*. Proses *training* dilakukan secara *sequential*, artinya data yang dimasukkan ke dalam proses *training* tidak langsung keseluruhan. Data awal dimasukkan per bagian dengan cara pembagian seperti yang sudah dijelaskan pada subbab sebelumnya. Selanjutnya, setiap kelompok data masuk ke proses *training*, pada saat proses *training* setiap kelompok data dilihat nilai *Mean Square Error* (MSE) setiap kelompok data.

Inisialisasi

Inisialisasi awal digunakan kelompok data pertama, dari perhitungan kelompok data pertama diperoleh w, b, β_0, dan MSE_0 sebagai inisialisasi bobot β_0 dan MSE awal. Bobot antara *input layer* dan *hidden layer* w diberikan secara random dengan ukuran matriks $\tilde{N} \times q$, dengan $\tilde{N}$ sebanyak *hidden node* dan q sebanyak atribut yang digunakan pada data. Dalam penelitian ini, *hidden node* yang digunakan adalah sebanyak N-1 dan atribut 2 maka w adalah matriks berukuran $(N-1) \times 2$. Sedangkan b berukuran $\tilde{N} \times 1$, dengan $\tilde{N}$ adalah sebanyak *hidden node* yang digunakan. Bobot β_0 diperoleh dari perkalian invers matriks *hidden layer* dengan target. Matriks *hidden layer* adalah matriks H seperti dirumuskan pada Persamaan 2.8 dengan ukuran dari matriks H adalah $N \times \tilde{N}$, dengan N adalah banyak data dan $\tilde{N}$ adalah banyak *hidden node*. Selanjutnya untuk perhitungan MSE_0 digunakan Persamaan 3.6.

$$w = [w_{i1},\ w_{i2}, \dots, w_{in}]^T$$

$$b = [w_1,\ w_2, \dots, w_n]^T$$

$$H_0 = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \cdots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix} \quad (4.1)$$

$$\beta_{(0)} = P_0 H_0^T T_0 \quad (4.2)$$

dengan

$$P_0 = \left(H_0^T H_0\right)^{-1} \quad (4.3)$$

T_0 adalah target kelompok data yang digunakan sebagai inisialisasi

Training* Integrasi Seleksi Data dan *Extreme Learning Machine

Setelah proses inisialisasi dengan kelompok data pertama selesai, dilanjutkan kelompok kedua masuk proses *training*. Setelah diperoleh bobot dan MSE, selanjutnya dicari nilai ΔMSE menggunakan Persamaan 3.1. Jika nilai $\Delta MSE \geq 0$ maka data tersebut digunakan sebagai *prototype* dan selanjutnya dilakukan update bobot β seperti pada Persamaan 3.7 sampai 3.9.

$$\beta_{n+1} = \beta_n + P_{n+1} H_{n+1}^T \left(T_{n+1} - H_{n+1}\beta_n\right)$$

dengan

$$P_{n+1} = P_n - P_n H_{n+1}^T (I + H_{n+1} P_n H_{n+1}^T)^{-1} H_{n+1} P_n$$

dan sebagai inisialisasi *P*

$$P_0 = \left(H_0{}^T H_0\right)^{-1}.$$

Keterangan:

$\beta^{(k+1)}$ = bobot β untuk data ke *k+1*

$\beta^{(k)}$ = bobot β untuk data ke *k*

H = matriks *hidden layer*

P_0 = inisalisasi P

Akan tetapi jika nilai $\Delta MSE < 0$ maka kelompok data tidak digunakan untuk data *training* dan tidak dilakukan update bobot seperti di atas.

4.2.4 *Testing* Integrasi Seleksi Data dan klasifikasi pada ELM

Proses *testing* merupakan proses untuk menguji seberapa baik performa dari suatu program yang telah dibuat. Pada proses ini modifikasi *Extreme Learning Machine*, yaitu integrasi antara proses seleksi data dan *Extreme Learning Machine* itu sendiri diuji. Langkah proses *testing* hampir sama dengan *training*, yang membedakan adalah, pada proses ini semua bobot diambil dari proses *training*, serta tidak ada proses seleksi data seperti pada saat *training*.

Pseudocode integrasi seleksi data dan *Extreme Learning Machine* (ELM) seperti pada Tabel 4.3.

Tabel 4.3 *Pseudocode* Integrasi seleksi data dan ELM

Pseudocode Integrasi seleksi data dan ELM	
1	$\aleph = \{(x_i, t_i) \mid x_i \in \boldsymbol{R^n}, t_i \in \boldsymbol{R^m},\ i \in [1, N]\}$
2	
3	Inisialisasi
4	M_0 ← Kelompok data pertama
5	T_0 ← Target kelompok data pertama
6	w ← matriks berukuran $\tilde{N} \times$ attribute yang elemenya
7	random
8	b ← matriks berukuran $\tilde{N} \times 1$ yang elemenya random
9	$temp_0 \leftarrow w_i x_j + b_i$
10	H_0 ← 1./(1+ exp(-$temp_0$))
11	P ← $(H_0^T H_0)^{-1}$
12	β_0 ← $P_0 H_0^T T_0$
13	Y_0 ← $H_0 \beta_0$
14	mse0 ← mse(T_0, Y_0)
15	k ← 0
16	
17	
18	Sequential Learning
19	for n == N0 : Block : n*Training*Data
20	M_{k+1} ← kelompok data ke k+1
21	T_{k+1} ← target kelompok data ke k+1
22	Temp ← $w_i x_j + b_i$
23	H_{k+1} ← 1./(1+exp(-temp))
24	cβ ← $(H_{k+1}^T H_{k+1}) H_{k+1}^T T_{k+1}$
25	Y ← H_{k+1} cβ
26	mse ← mse(T_{k+1}, Y)
27	delta← mse-mse0
28	
29	if delta<0
30	mse=mse0
31	else
32	$P_{k+1} \leftarrow P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k$
33	$\beta_{k+1} \leftarrow \beta_k + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta_k)$
34	mse0=mse
35	end
36	end
37	
38	

4.2.5 Menghitung Akurasi

Karena data *binding site* protein merupakan data yang mempunyai karakteristik *imbalanced*, artinya banyak data positif (*binding site*) dan negatif (*non binding site*) mempunyai perbandingan yang jauh. Sehingga untuk mengukur performa dari program tidak dapat menggunakan perhitungan akurasi biasa, tetapi penggunakan akurasi, *precision*, *recall*, *specificity*, dan *G-mean*.

Misalkan dalam sebuah *testing* dengan target:

T = [1 0 0 0 1 1 1 1 0 0]

sedangkan hasil klasifikasinya adalah:

Y = [1 1 0 0 0 1 1 0 1 0]

sehingga *confusion matrix* dari hasil di atas dapat dilihat pada Tabel 4.4

Tabel 4.4 Contoh *Confusion Matrix*

		Nilai Sebenarnya	
		True	False
Prediksi	True	3	2
	False	2	3

Dari hasil di atas dapat diperoleh nilai TP= 3 karena data yang target 1 dikenali dengan 1 ada 3. Nilai FN= 2 katena data dengan target 1 dikenali 0 ada 2. Nilai TN=3 karena data dengan target 0 dikenali 0 ada 3. Sedangkan nilai FP=2 karena data dengan target 0 dikenali dengan satu ada 2 data. Dari *confusion matrix* dapat dicari nilai akurasi *precision*, *recall*, dan *G-mean*.

$$\text{Akurasi} = \frac{TP + TN}{\text{total sampel}} = \frac{3+3}{10} = 0.6$$

$$precision = \frac{TP}{TP + FP} = \frac{3}{3+2} = 0.6$$

$$recall = \frac{TP}{TP + FN} = \frac{3}{3+2} = 0.6$$

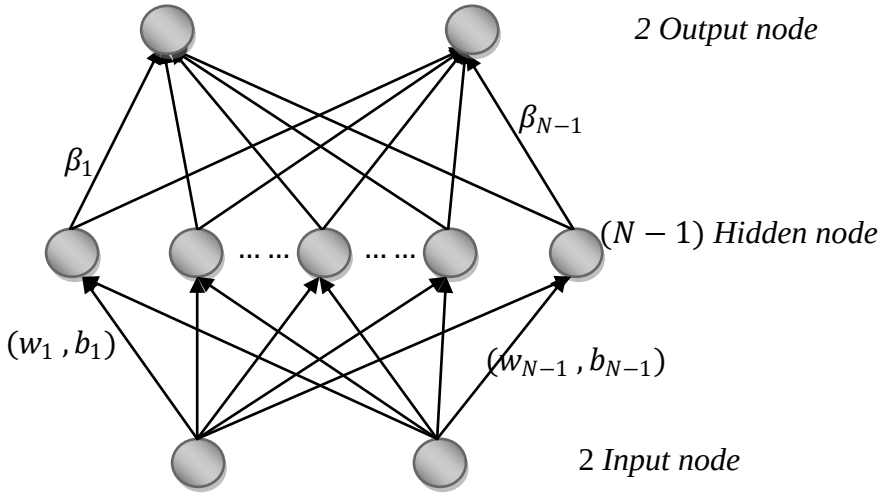
$$specificity = 1 - \frac{FP}{FP + TN} = 1 - \frac{2}{2+3} = 0.6$$

$$\text{G-}mean = \sqrt{sensitivity \times specificity} = \sqrt{0.6 \times 0.6} = 0.6$$

dari perhitungan di atas diperoleh hasil akurasi, *precision*,*recall*, *specificity* dan *G-mean* sama yaitu 0.6 atau 60%.

4.3 Prediksi protein 2ZAL dengan data *training* 1YBU

Artsitektur jaringan IDELM untuk prediksi *binding site* protein-ligan dapat dilihat pada Gambar 4.2.



Gambar 4.2 Arsitektur jaringan IDELM untuk prediksi *binding site* protein-ligan

Dari Gambar 4.2 dapat dilihat bahwa pada asitektur jaringan IDELM untuk prediksi *binding site* protein-ligan terdapat 2 *node* input pada *input layer*. 2 *node* pada input layer menunjukkan banyak atribut yang digunakan dalam prediksi *binding site* protein-ligan dalam penelitian ini ada 2 yaitu jarak titik grid dengan atom ligan terdekat dan skor titik grid. Sebagaimana dijelaskan pada bab 3 dalam penelitian ini *hidden node* yang digunakan sebanyak N-1 *node*, dengan N adalah banyak data yang masuk proses *training*. Setelah melalui hidden layer selanjutnya untuk mendapatkan bobot β digunakan Persamaan 2.8. Pada arsitektur jaringan IDELM terdapat 2 *node* pada *output layer* karena dalam penelitian ini IDELM digunakan untuk klasifikasi biner, mempunyai dua target yaitu 1 dan 0. Pada prediksi *binding site* protein-ligan, 1 adalah target untuk binding site, dan 0 adalah target untuk *non binding site*.

Untuk contoh studi kasus dalam pembahasan digunakan protein dengan PDB ID 2ZAL yang merupakan protein dari kelompok *hydrolase* dengan data *training* nya adalah protein dengan PDB ID 1YBU yang juga merupakan protein dari kelompok *hydrolase*. Proses prediksi dilakukan dengan memasukkan PDB ID dari data yang akan diprediksi *binding site* nya. Setelah proses *training* selesai, selanjutnya plot *binding site* muncul pada grafik pertama dan grafik kedua, grafik "*Predicting Binding Site*" adalah plot koordinat 3D hasil dari klasifikasi,

sedangkan grafik "*Actual Binding Site*" adalah plot koordinat 3D *binding site* yang sebenarnya.

Data protein 2ZAL dan 1YBU diambil dari *webserver* RCSB Protein Data Bank, selanjutnya data yang berupa file pdb di analisis menggunakan webserver LISE untuk mendapatkan *scor conservation* serta hasil analisis geometri dan energi yang berupa skor titik grid dan jarak titik grid ke atom ligan terdekat. Data hasil analisis dari LISE masih berupa file pdb, akan tetapi dalam penelitian ini hanya diambil koordinat tiga dimensi dan hasil perhitungan jarak titik grid ke atom ligan terdekat dan skor setiap titik grid. Data koordinat tiga dimensi dan hasil perhitungan LISE dituliskan pada Tabel 4.5.

Tabel 4.5 Data koordinat 3D, jarak titik grid ke atom ligan terdekat, dan *Grid score* protein 1YBU dan titik grid 3D yang berpotensi sebagai binding site

Koordinat			Jarak titik grid ke atom ligan terdekat	*Grid score*	Target
x	y	z			
36	63	61	4.36	60.33	1
36	64	54	5.39	33.34	1
36	64	55	4.69	33.93	1
36	64	56	4.12	35.32	1
36	64	57	3.74	38.69	1
36	64	58	3.61	42.61	1
36	64	59	3.74	47.83	1
36	64	60	4.12	53.75	1
36	65	57	4.36	38.27	1
37	60	61	5.39	60.08	1
37	61	55	5.1	39.33	1
37	61	56	4.58	38.85	1
37	61	57	4.24	40.17	1
:	:	:	:	:	:
45.583	55.079	68.248	0.5	74.43	0
24.372	48.188	40.87	1	66.61	0
31.956	26.165	68.179	1	26.22	0
30.419	61.968	75.737	1	44.26	0
:	:	:	:	:	:
51.293	44.67	58.344	1	60.98	0

Dari data pada Tabel 4.6 data yang digunakan untuk *training* hanya data jarak titik grid ke atom ligan terdekat dan *Grid score*. Sebelum data digunakan,

terlebih dahulu data *Score Conservation*, dan *Grid score* dari masing-masing data dinormalisasi dengan menggunakan rumus minmax.

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Untuk data 1YBU pada data nilai $x_{min} = 0$ sedangkan x_{max} = 99.93, sehingga untuk data ternormalisasi pada protein YBU dirumuskan sebagai berikut:

$$x_{i_{normalized}} = \frac{x_i - 0}{99.93 - 0}.$$

Sehingga diperoleh data 1YBU yang ternormalisasi pada Tabel 4.7.

Tabel 4.5 Data koordinat 3D, *Jarak titik grid ke atom ligan terdekat*, dan *Grid score* protein 2ZAL dan titik grid 3D yang berpotensi sebagai binding site

Koordinat			*Score Conservation*	*Grid score*	Target
X	y	z			
3	-4	44	5.48	36.02	1
4	-6	44	5.39	18.63	1
4	-6	45	5.1	19.34	1
4	-6	46	5	21.8	1
4	-5	43	5.39	26.25	1
4	-5	44	4.9	26.59	1
4	-5	45	4.58	27.5	1
4	-4	43	5.1	34.57	1
:	:	:	:	:	:
:	:	:	:	:	:
8	7	82	4.69	39.96	1
8	7	83	4.12	39.05	1
8	7	84	3.74	38.96	1
8	7	85	3.61	38.55	1
8	8	83	3.74	33.38	1
32.142	3.159	58.605	1	35.8	0
20.182	-7.67	57.427	1	36.67	0
-8.317	-7.413	46.395	1	41.16	0
35.302	1.435	64.212	0.5	27.72	0
:	:	:	:	:	:
:	:	:	:	:	:
32.301	14.539	48.737	1	37.83	0

Sedangkan untuk data 2ZAL pada data nilai $x_{min} = 0$ sedangkan x_{max} = 99.82, sehingga untuk data ternormalisasi pada protein YBU ditumuskan sebagai berikut:

$$x_{i_{normalized}} = \frac{x_i - 0}{99.82 - 0}$$

Sehingga diperoleh data 2ZAL pada Tabel 4.8.

Tabel 4.7 Hasil normalisasi data protein 1YBU

Koordinat			*Score Conservation*	*Grid score*	Target
X	y	Z			
36	63	61	0.04363	0.60372	1
36	64	54	0.05394	0.33363	1
36	64	55	0.04693	0.33954	1
36	64	56	0.04123	0.35345	1
36	64	57	0.03743	0.38717	1
36	64	58	0.03613	0.42640	1
36	64	59	0.03743	0.47864	1
36	64	60	0.04123	0.53788	1
36	65	57	0.04363	0.38297	1
37	60	61	0.05394	0.60122	1
37	61	55	0.05104	0.39358	1
37	61	56	0.04583	0.38877	1
37	61	57	0.04243	0.40198	1
:	:	:	:	:	:
:	:	:	:	:	:
45.583	55.079	68.248	0.00500	0.74482	0
24.372	48.188	40.87	0.01001	0.66657	0
31.956	26.165	68.179	0.01001	0.26238	0
30.419	61.968	75.737	0.01001	0.44291	0
:	:	:	:	:	:
51.293	44.67	58.344	0.01001	0.61023	0

a. Proses *Training*

Setelah proses normalisasi, selanjutnya data masuk ke proses *training* data. Dalam proses *training*, data masuk dilakukan secara *sequential* 100-by-100. Inisialisasi bobot dan MSE menggunakan kelompok data pertama, yaitu 100 data pertama. Misalkan:

M_i = data kelompok ke t-1

M_0 = inisialisasi data awal yaitu data kelompok pertama

t_0 = data target kelompok pertama

MSE_0 = *Mean Square Error* kelompok data pertama

w = bobot antara *input layer* dan *hidden layer*

b = bobot bias

β_0 = bobot antara *hidden layer* dan *output layer* kelompok pertama

Tabel 4.8 Hasil normalisasi data protein 2ZAL

Koordinat			*Score Conservation*	*Grid score*	Target
X	y	Z			
3	-4	44	0.054893	0.360813	1
4	-6	44	0.053992	0.186617	1
4	-6	45	0.051087	0.193729	1
4	-6	46	0.050085	0.218371	1
4	-5	43	0.053992	0.262947	1
4	-5	44	0.049083	0.266353	1
4	-5	45	0.045878	0.275468	1
4	-4	43	0.051087	0.346289	1
:	:	:	:	:	:
:	:	:	:	:	:
8	7	82	0.04698	0.40028	1
8	7	83	0.04127	0.391165	1
8	7	84	0.037464	0.390263	1
8	7	85	0.036161	0.386156	1
8	8	83	0.037464	0.334368	1
32.142	3.159	58.605	0.010017	0.35861	0
20.182	-7.67	57.427	0.010017	0.367324	0
-8.317	-7.413	46.395	0.010017	0.412301	0
35.302	1.435	64.212	0.005009	0.277672	0
:	:	:	:	:	:
:	:	:	:	:	:
32.301	14.539	48.737	0.010017	0.378944	0

Inisialisasi

Dalam penetian ini *hidden node* yang digunakan sebanyak $\widetilde{N} = N - 1$, diambil *hidden node* sebanyak $N - 1$, karena dari bebrapa penelitian sebelumnya untuk mendapatkan banyak *hidden node* yang digunakan biasanya adalah sebanyak N atau sebanyak data yang digunakan sebagaimana yang dilakukan oleh Huang (2008), Hidayatullah (2014). Penggunaan *hidden node* sebanyak N ini bertujuan untuk mendapatkan nilai error *training* 0, akan tetapi error *training* 0 tidak menjamin performa terbaik dari ELM (Huang, 2012). Sedangkan, pada penelitian

yang lain banyak *hidden node* diberikan dengan *trial error* untuk mendapatkan hasil yang optimal sebagaimana yang dilakukan oleh beberapa peneliti sebelumnya, diantaranya adalah Zhu (2005), Liang (2006), dan Lou (2015). Karena dalam penelitian ini dilakukan seleksi data maka banyak *hidden node* dipilih $\tilde{N} = N - 1$. Sehingga, pada kelompok data pertama *hidden node* nya sebanyak 99 tentukan bobot w dan bobot bias b secara random, dalam hal ini ukuran dari $w$adalah 99× 2 dan b merupakan matriks berukuran 99×1.

$$w = \begin{bmatrix} -0.7146 & 0.1902 & -0.2511 & \cdots & 0.6944 & 0.8821 & -0.2609 & -0.9860 \\ 0.9125 & 0.1630 & -0.0003 & \cdots & -0.4690 & -0.0037 & -0.3388 & -0.8710 \end{bmatrix}^T$$

$$b = \begin{bmatrix} 0.7096 & 0.1485 & 0.2340 & 0.8713 & \cdots & 0.3276 & 0.4182 & 0.8905 \end{bmatrix}^T$$

Setelah ditentukan w dan b, selanjutnya dicari nilai matriks H dengan rumus sebagai berikut:

$$H = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \cdots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}$$

Sehingga diperoleh:

$$H = \begin{bmatrix} 0.8091 & 0.5737 & 0.5548 & 0.8474 & 0.8474 & 0.3998 & 0.8008 & 0.7278 & \cdots & 0.5248 \\ 0.7997 & 0.5711 & 0.5548 & 0.8389 & 0.5827 & 0.4155 & 0.7950 & 0.7179 & \cdots & 0.5387 \\ 0.7896 & 0.5681 & 0.5550 & 0.8292 & 0.5881 & 0.4324 & 0.7894 & 0.7061 & \cdots & 0.5549 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.7195 & 0.5503 & 0.5559 & 0.7605 & 0.6217 & 0.5375 & 0.7510 & 0.6309 & \cdots & 0.6471 \end{bmatrix}$$

Hitung bobot β

$$\beta = H^{\dagger}T$$

$H^{\dagger}$ adalah *Moore Penrose Generalized Inverse* yang dapat diselesaikan dengan rumus berikut:

$$H^{\dagger} = P_0 H^T$$

$$P_0 = \left(H_0{}^T H_0\right)^{-1}.$$

sehingga diperoleh

$$\beta = \begin{bmatrix} 0.0118 \\ 0.0665 \\ 0.0629 \\ \vdots \\ 0.0045 \end{bmatrix}$$

Setelah mendapatkan bobot β selanjutnya dihitung *output*.

$$Y = H\beta$$

$$Y = \begin{bmatrix} 0.8091 & 0.5737 & 0.5548 & 0.8474 & 0.8474 & 0.3998 & 0.8008 & 0.7278 & \cdots & 0.5248 \\ 0.7997 & 0.5711 & 0.5548 & 0.8389 & 0.5827 & 0.4155 & 0.7950 & 0.7179 & \cdots & 0.5387 \\ 0.7896 & 0.5681 & 0.5550 & 0.8292 & 0.5881 & 0.4324 & 0.7894 & 0.7061 & \cdots & 0.5549 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.7195 & 0.5503 & 0.5559 & 0.7605 & 0.6217 & 0.5375 & 0.7510 & 0.6309 & \cdots & 0.6471 \end{bmatrix} \begin{bmatrix} 0.0118 \\ 0.0665 \\ 0.0629 \\ \vdots \\ 0.0045 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0.9997 \\ 0.9997 \\ 0.9997 \\ \vdots \\ 0.9997 \end{bmatrix}$$

Selanjutnya hitung MSE data kelompok pertama

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

$$MSE_0 = \frac{(0.9997\text{-}1)^2 + (0.9997\text{-}1)^2 + (0.9997\text{-}1)^2 + ... + (0.9997\text{-}1)^2}{100}$$

$$= \frac{\left(-3\times10^{-4}\right)^2 + \left(-3\times10^{-4}\right)^2 + \left(-3\times10^{-4}\right)^2 + ... + \left(-3\times10^{-4}\right)^2}{100} = \left(-3\times10^{-4}\right)^2$$

Perhitungan kelompok data kedua

Data *input* pada perhitungan selanjutnya adalah kelompok data kedua, yaitu 100 data kedua dari data. Sehingga pada kelompok data kedua *hidden node* nya tetap sebanyak 99 tentukan vektor *w* dan vektpr *b* tetap seperti matriks *w* dan vektpr *b* yang digunakan pada data kelompok pertama, dalam hal ini ukuran dari *w* adalah 99× 2 dan *b* merupakan vektor berukuran 99×1. Dengan langkah yang seperti sebelumnya diperoleh matriks H.

$$H = \begin{bmatrix} 0.5874 & 0.6307 & 0.6569 & 0.6598 & 0.5671 & 0.6847 & 0.4695 & 0.6660 & \cdots & 0.5138 \\ 0.5888 & 0.6283 & 0.6539 & 0.6587 & 0.5664 & 0.6832 & 0.4711 & 0.6652 & \cdots & 0.5161 \\ 0.5838 & 0.7178 & 0.7478 & 0.6375 & 0.6636 & 0.7052 & 0.3668 & 0.6791 & \cdots & 0.4195 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5850 & 0.7028 & 0.7322 & 0.6410 & 0.6473 & 0.7012 & 0.3846 & 0.6766 & \cdots & 0.4365 \end{bmatrix}$$

Dengan cara yang sama pula pada proses inisialisasi diperoleh bobot β dan Y.

$$\beta = \begin{bmatrix} 0.0272 \\ -0.0029 \\ -0.0083 \\ \vdots \\ 0.0473 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1.0001 \\ 1.0001 \\ 1.0001 \\ \vdots \\ 1.0001 \end{bmatrix}$$

Sehingga nilai MSE nya adalah sebagai berikut:

$$MSE_1 = \frac{(1.0001\text{-}1)^2 + (1.0001\text{-}1)^2 + (1.0001\text{-}1)^2 + ... + (1.0001\text{-}1)^2}{100}$$

$$= \frac{(1\times10^{-4})^2 + (1\times10^{-4})^2 + (1\times10^{-4})^2 + ... + (1\times10^{-4})^2}{100} = \left(1\times10^{-4}\right)^2.$$

Karena nilai MSE_0 lebih besar dari MSE_1, artinya nilai dari $\Delta MSE < 0$ maka data kelompok kedua tidak digunakan sebagai data *prototype*, dalam artian kelompok data kedua dihapus dan tidak dilakukan update bobot.

Langkah yang sama seperti di atas dilakukan juga untuk data kelompok ketiga dan seterusnya sampai kelompok data habis. Dengan ketentuan, jika $\Delta MSE < 0$ maka kelompok data dihapus dan tidak ada proses update bobot. Pada saat $\Delta MSE \geq 0$ maka kelompok data digunakan sebagai data *prototype* dan dilakukan update bobot.

$$\beta_{n+1} = \beta_n + P_{n+1} H_{n+1}^T \left(T_{n+1} - H_{n+1}\beta_n\right)$$

dengan

$$P_{n+1} = P_n - P_n H_{n+1}^T (I + H_{n+1} P_n H_{n+1}^T)^{-1} H_{n+1} P_n$$

b. Proses *Testing*

Dalam penelitian ini proses *testing* juga dilakukan secara *sequential* dengan langkah yang sama dengan proses *testing*. Akan tetapi, pada proses *testing* seluruh bobot diambil dari hasil *training*. Dari hasil *testing* protein 2ZAL diperoleh hasil klasifikasi. Selanjutnya dilakukan ploting, jika data 2ZAL diklasifikasikan sebagai 1 maka koordinat tiga dimensi dari data yang diklasifikasikan sebagai 1 adalah titik grid yang berpotensi sebagai binding site, sedangkan data yang diklasifikasikan sebagai 0 adalah titik grid yang bukan binding site. Dari hasil *testing* diperoleh *output* (lampiran G), selanjutnya dibuat *confusion matriks* sebagai berikut:

Tabel 4.9 *Confusion Matriks testing* protein 2ZAL

		Nilai Sebenarnya	
		True	False
Prediksi	True	600	0
	False	88	4598

Dari hasil di atas dapat diperoleh nilai TP= 600 karena data yang target 1 dikenali dengan 1 ada 600. Nilai FN= 88 karena data dengan target 1 dikenali 0 ada 88. Nilai TN= 4598 karena data dengan target 0 dikenali 0 ada 4598. Sedangkan nilai FP=0 karena tidak data dengan target 0 dikenali dengan 1. Dari *confusion matrix* dapat dicari nilai akurasi, *recall*, dan *G-mean* nya.

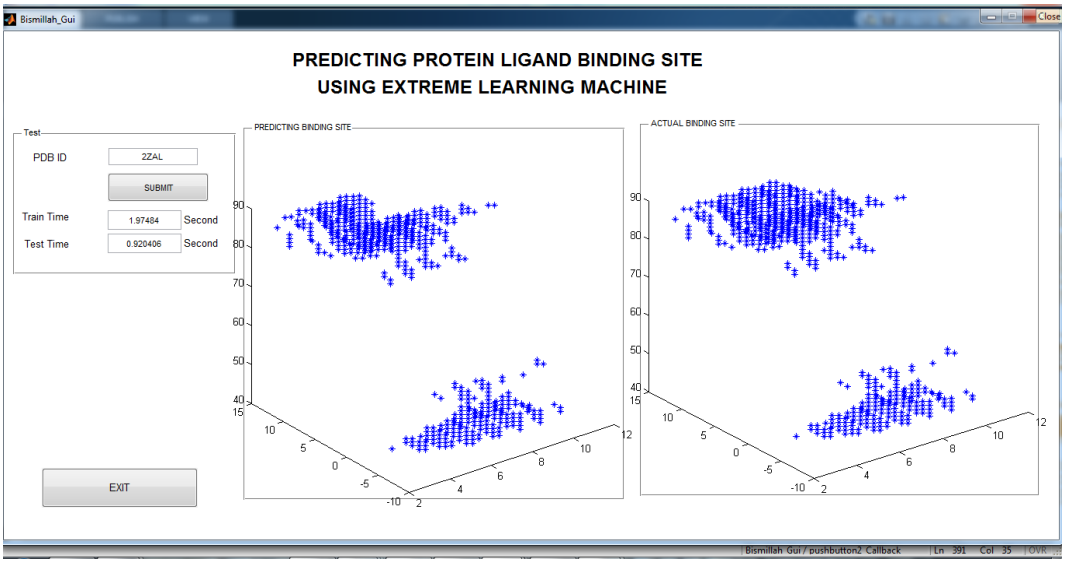
$$\text{akurasi} = \frac{TP + TN}{\text{total sampel}} = \frac{600 + 4598}{5286} = 0.9834$$

$$recall = \frac{TP}{TP + FN} = \frac{600}{600 + 88} = 0.8721$$

$$Specificity = 1 - \frac{FP}{FP + TN} = 1 - \frac{0}{0 + 4598} = 1$$

$$G\text{-}mean = \sqrt{sensitivity \times specificity} = \sqrt{0.8721 \times 1} = 0.9339$$

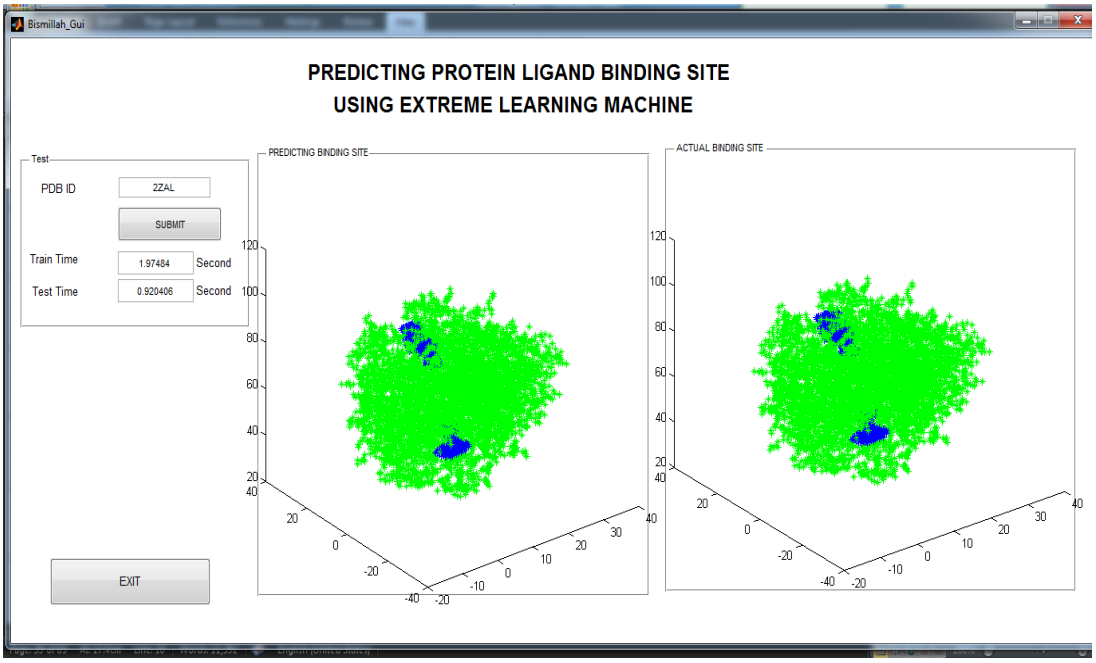
Selanjutnya, koordinat tiga dimensi diplot berdasarkan hasil klasifikasi. Hasil prediksi *binding site* protein-ligan untuk data 2ZAL pada GUI dengan IDELM dapat dilihat pada Gambar 4.3 sampai Gambar 4.7.



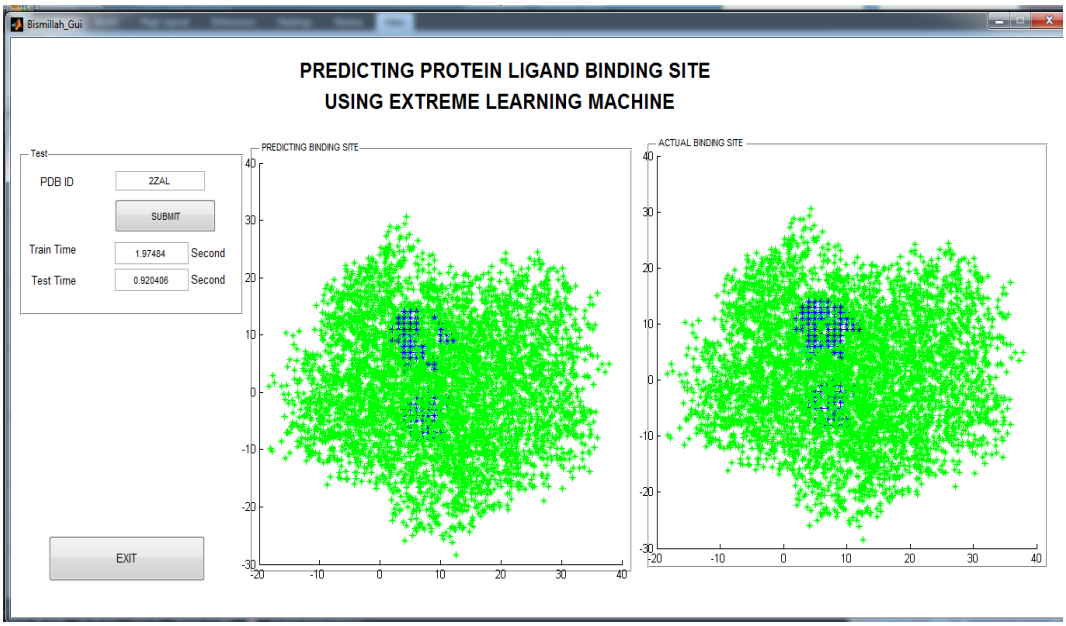
Gambar 4.3 Tampilan hasil running program dengan GUI Matlab menampilkan *Binding Site* 2ZAL

Gambar 4.3 menunjukkan hasil running program yaitu prediksi *binding site* protein-ligan. Plot di atas adalah plot dari koordinat *binding site*. Plot sebelah kiri menunjukkan hasil prediksi *binding site* protein-ligan dengan menggunakan integrasi seleksi data dan ELM. Dari kedua plot tesebut dapat dibandingkan antara hasil prediksi *binding site* protein-ligan dan *binding site* yang sebenarnya.

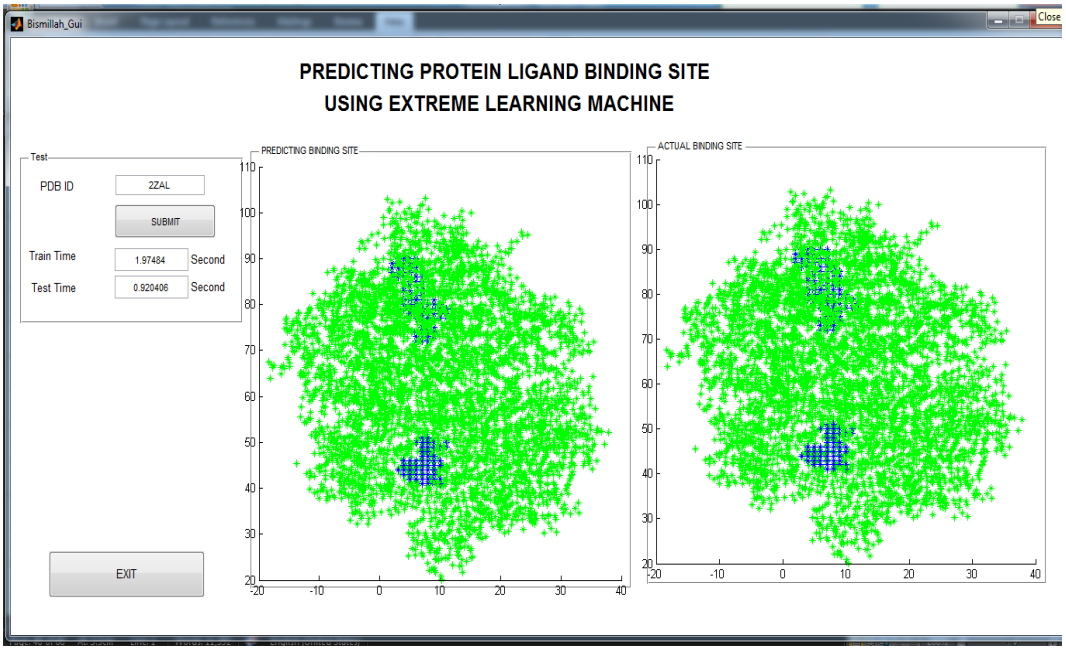
Dari Gambar 4.4 plot atau titik-titik yang berwarna hijau merupakan plot dari protein dan selain dari binding site. Plot sebelah kiri menunjukkan plot hasil prediksi dengan menggunakan IDELM, sedangkan plot sebelah kanan merupakan keadaan sebenarnya dari protein dan binding site. Pada Gambar 4.4 tidak terlalu tampak perbedaan dari hasil prediksi dan keadaan sebenarnya dari protein dan binding site. Pada Gambar 4.5, Gambar 4.6 dan Gambar 4.7 mulai terlihat perbedaan dari plot hasil prediksi dan keadaan sebenarnya. Gambar 4.5 merupakan plot koordinat tiga dimensi yang tampak dari sumbu Z. Gambar 4.6 merupakan plot koordinat tiga dimensi yang tampak dari sumbu Y. Sedangkan, Gambar 4.7 merupakan plot koordinat tiga dimensi yang tampak dari sumbu X.



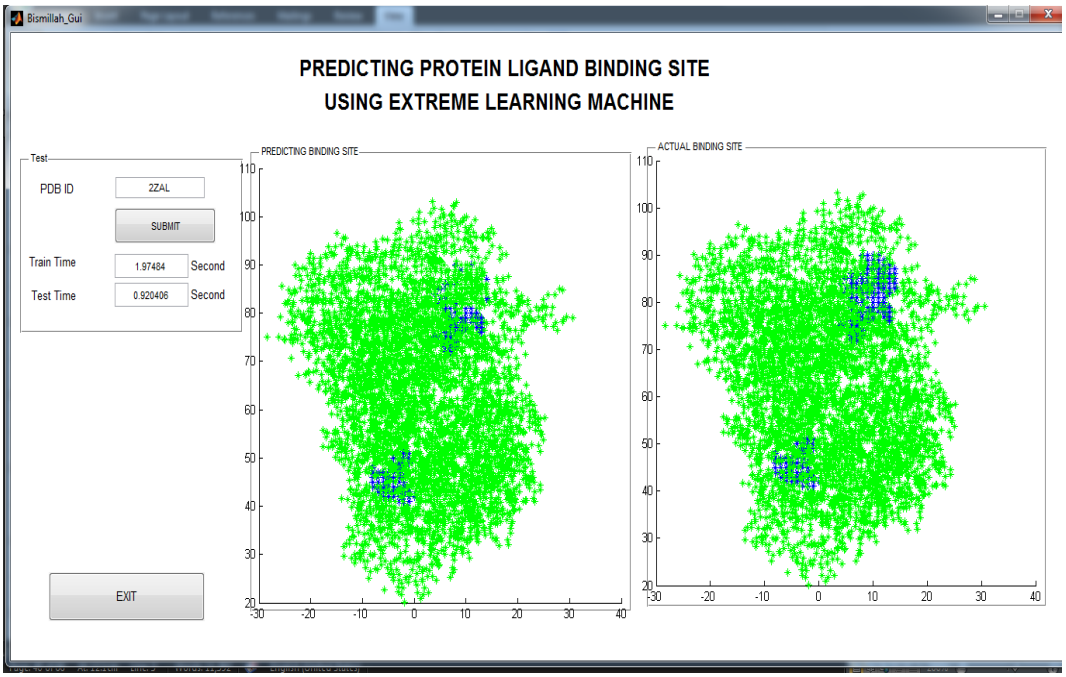
Gambar 4.4 Tampilan hasil running program dengan GUI Matlab menampilkan *Binding Site* dan protein 2ZAL dalam bentuk 3 dimensi



Gambar 4.5 Tampilan hasil running program dengan GUI Matlab menampilkan *Binding Site* dan protein 2ZAL tampak dari sumbu Z



Gambar 4.6 Tampilan hasil running program dengan GUI Matlab menampilkan *Binding Site* dan protein 2ZAL tampak dari sumbu Y



Gambar 4.7 Tampilan hasil running program dengan GUI Matlab menampilkan *Binding Site* dan protein 2ZAL tampak dari sumbu X

4.4 Performa Integrasi Seleksi Data dan ELM dengan Uji *Benchmark Data* dan untuk Prediksi *Binding site* Protein-Ligan

Sebelum diterapkan untuk prediksi *binding site* protein-ligan, algoritma yang digunakan dalam penelitian ini yaitu integrasi seleksi data dan klasifikasi pada *Extreme Learning Machine* diuji terlebih dahulu dengan menggunakan data dari UCI *Machine Learning* dan *Benchmark Data Sets for Highly Imbalanced Binary Classification* (Ding, 2011) yang mempunyai karakter data yang *imbalanced* .

4.4.1 Performa Integrasi Seleksi Data dan Klasifikasi pada ELM dengan Benchmark Data Sets

Dalam pengujian performa integrasi seleksi data dan klasifikasi pada ELM dengan *Benchmark data* digunakan 5-*fold cross validation*, dengan kata lain 80% data *training* dan 20% data *testing* dilakukan lima kali *training*. Untuk pengujian performa integrasi seleksi data dan klasifikasi pada ELM menggunakan 13 data yang *imbalanced* , dan dibandingkan dengan beberapa algoritma lain, yaitu ELM standar, Backpropagation, dan SVM. Karena merupakan data *imbalanced* untuk performa algoritma yang digunakan tidak hanya dilihat dari akurasi biasa, tetapi juga dari *precision*, *recall*, dan *G-mean*.

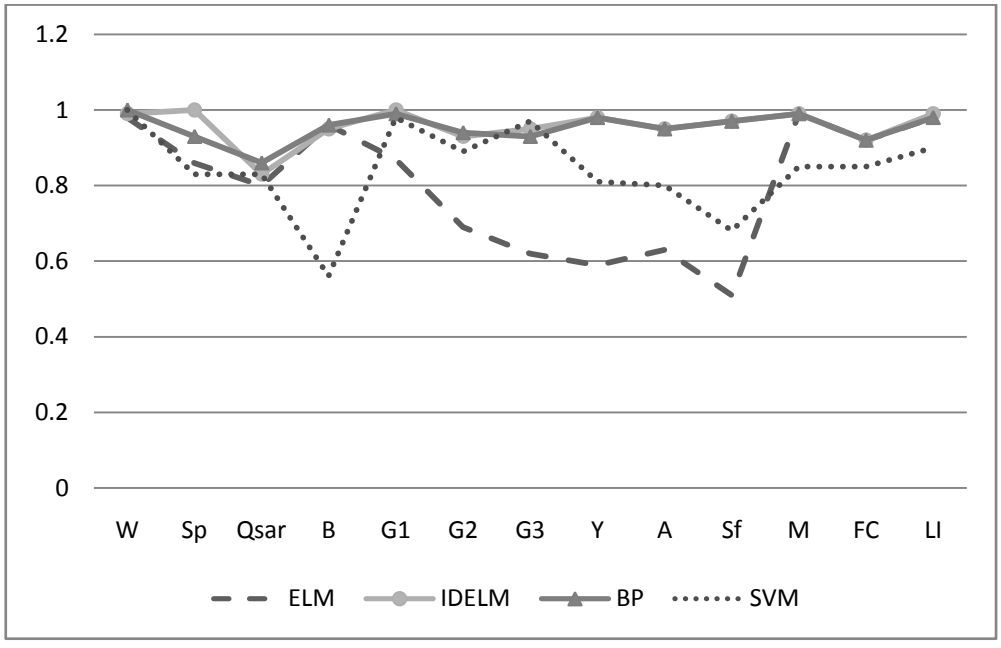
a. Akurasi

Tabel akurasi Integrasi seleksi data dan klasifikasi pada ELM (IDELM) dibandingkan dengan ELM, Backpropagation (BP), dan SVM menggunakan *benchmark data* dapat dilihat pada Tabel 4.10.

Dari Tabel 4.10 dapat dilihat bahwa integrasi seleksi data dan klasifikasi ELM dapat meningkatkan performa dari algoritma ELM, yang sebelumnya rata-rata akurasi dari ELM adalah 0.8 atau 80% menjadi 95,76%. Selain itu integrasi seleksi data dan klasifikasi ELM mempunyai akurasi rata-rata yang lebih baik dari Backpropagation dengan selisih 0.48% lebih baik IDELM. Untuk lebih jelas hasil akurasi di atas dapat ditampilkan dalam bentuk grafik Pada Gambar 4.8.

Tabel 4.10 Akurasi dari uji menggunakan *benchmark data*

Data	**Akurasi**			
	ELM	**IDELM**	**BP**	**SVM**
Wilt (W)	0.98	0.99	1.00	1.00
Spambase(Sp)	0.86	1.00	0.93	0.83
Qsar	0.80	0.83	0.86	0.83
Balance(B)	0.96	0.95	0.96	0.56
Glass1(G1)	0.87	1.00	0.99	0.98
Glass2(G2)	0.69	0.93	0.94	0.89
Glass3(G3)	0.62	0.95	0.93	0.97
Yeast(Y)	0.59	0.98	0.98	0.81
Abalone(Y)	0.63	0.95	0.95	0.80
Solar Flare(SF)	0.51	0.97	0.97	0.68
Mammography(M)	0.99	0.99	0.99	0.85
Forest Cover(FC)	0.92	0.92	0.92	0.85
Letter Img(LI)	0.98	0.99	0.98	0.9
Rata-rata	0.8	0.9576	0.9538	0.8415



Gambar 4.8 Gafik akurasi dari uji menggunakan *benchmark data*

Dari Gambar 4.8 dapat dilihat bahwa integrasi seleksi data dan klasifikasi hampir selalu sama akurasinya, kecuali pada data *spambase* akurasi IDLM lebih baik dari Backpropagation. Akurasi SVM hampir sama dengan IDELM dan BP pada data Glass 1, Glass2, dan Glass3, karena pada data ini perbandingan data positif dan negatifnya tidak terlalu jauh, sehingga SVM dapat memiliki akurasi yang hampir sama. Data di atas adalah merupakan data yang *imbalanced* sehingga akurasi biasa seperti di atas tidak dapat digunakan untuk mengukur dengan baik performa dari masing-masing algoritma yang digunakan.

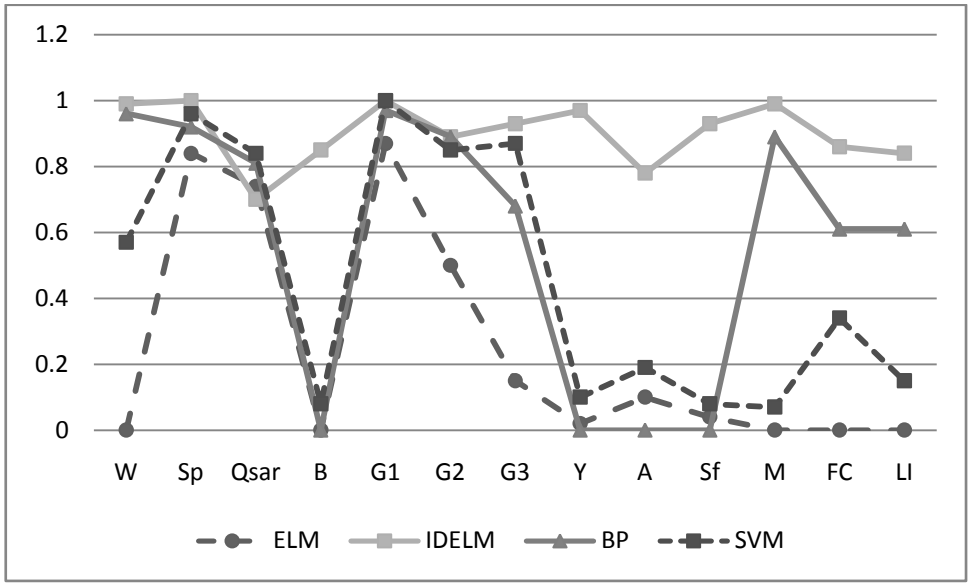
b. *Precision*

Tabel *precision* Integrasi seleksi data dan klasifikasi pada ELM (IDELM) dibandingkan dengan ELM, Backpropagation (BP), dan SVM menggunakan *benchmark data* ditampilkan pada Tabel 4.11.

Tabel 4.11 *Precision* dari uji menggunakan *benchmark data*

Data	***Precision***			
	ELM	**IDELM**	**BP**	**SVM**
Wilt (W)	0	0.99	0.96	0.57
Spambase(Sp)	0.84	1.00	0.92	0.96
Qsar	0.74	0.70	0.81	0.84
Balance(B)	0	0.85	0	0.08
Glass1(G1)	0.87	1.00	0.97	1.00
Glass2(G2)	0.50	0.89	0.89	0.85
Glass3(G3)	0.15	0.93	0.68	0.87
Yeast(Y)	0.02	0.97	0	0.10
Abalone(Y)	0.10	0.78	0	0.19
Solar Flare(SF)	0.04	0.93	0	0.08
Mammography(M)	0	0.99	0.89	0.07
Forest Cover(FC)	0	0.86	0.61	0.34
Letter Img(LI)	0	0.84	0.61	0.15
Rata-rata	0.2507	0.9023	0.5815	0.4692

Precision adalah porsi dari prediksi data positif yang dikenali benar oleh algoritma. Dari Tabel 4.11 dapat dilihat bahwa IDLM mempunyai nilai rata-rata *precision* paling baik dibandingkan ELM, BP, dan SVM, yaitu 0.9023 atau 90.23% dari data positif. Untuk lebih jelas melihat perbandingan *precision* setiap data, dapat dilihat Gambar 4.9.



Gambar 4.9 Grafik *precision* dari uji menggunakan *benchmark data*

Dari Gambar 4.9 terlihat bahwa *precision* dari setiap data untuk ELM, BP dan SVM tidak stabil, artinya pada beberapa data nilai *precision* dari ketiga algorima tersebut kadang tinggi dan kadang rendah bahkan bias sampai nilai 0. Hal tersebut dipengaruhi oleh banyak data, perbandingan data positif dan negatif yang besar, dan banyak atribut yang digunakan.

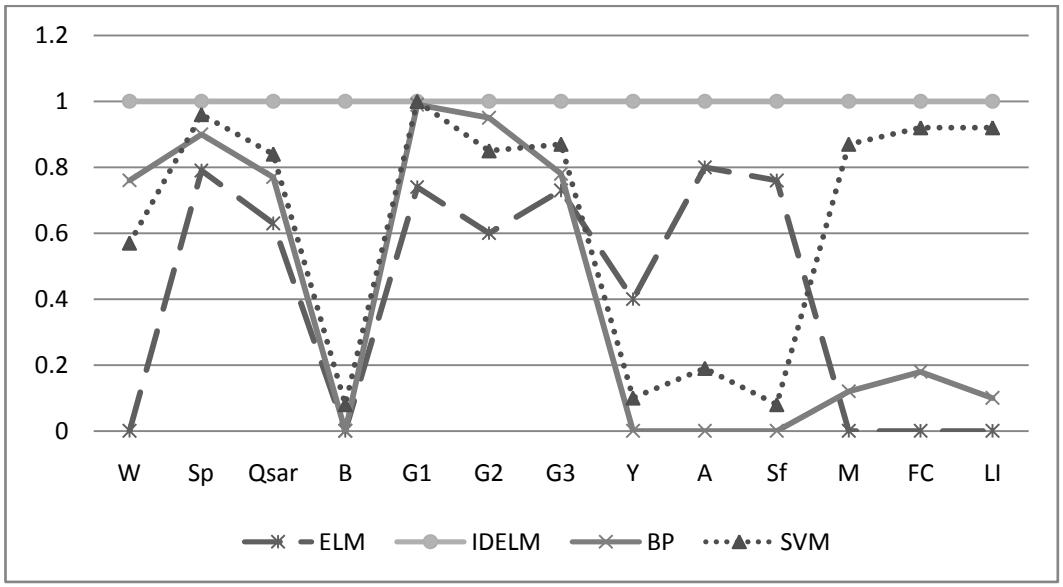
c. *Recall*

Tabel *recall* Integrasi seleksi data dan klasifikasi pada ELM (IDELM) dibandingkan dengan ELM, Backpropagation (BP), dan SVM menggunakan *benchmark data* dapat dilihat pada Tabel 4.12.

Recall adalah porsi dari data sampel yang diprediksi benar oleh algoritma. Dari Tabel 4.12 dapat dilihat bahwa IDELM mempunyai nilai *recall* 1 pada semua data. Sedangkan untuk ELM, BP, dan SVM rata-rata *recall* nya masing-masing adalah 0.4192, 0.4269, dan 0.6346. Untuk lebih jelas melihat perbandingan *recall* setiap data, dapat dilihat pada Gambar 4.10.

Tabel 4.12 *Recall* dari uji menggunakan *benchmark data*

Data	***Recall***			
	ELM	**IDELM**	**BP**	**SVM**
Wilt (W)	0	1	0.76	0.57
Spambase(Sp)	0.79	1	0.9	0.96
Qsar	0.63	1	0.77	0.84
Balance(B)	0	1	0	0.08
Glass1(G1)	0.74	1	0.99	1.00
Glass2(G2)	0.60	1	0.95	0.85
Glass3(G3)	0.73	1	0.78	0.87
Yeast(Y)	0.40	1	0	0.10
Abalone(Y)	0.80	1	0	0.19
Solar Flare(SF)	0.76	1	0	0.08
Mammography(M)	0	1	0.12	0.87
Forest Cover(FC)	0	1	0.18	0.92
Letter Img(LI)	0	1	0.1	0.92
Rata-rata	0.4192	1	0.4269	0.6346



Gambar 4.10 Grafik *recall* dari uji menggunakan *benchmark data*

Sama halnya dengan *precision, recall* untuk *benmarck data*, IDLM mempunyai nilai *recall* yang tetap yaitu 1, artinya semua data positif pada setiap data terklasifikasi dengan benar oleh IDELM. Sedangkan ELM, BP dan SVM tergantung pada jumlah data, banyak atribut yang digunakan, dan perbandingan data *imbalanced* nya. Pada ELM dan BP ada beberapa data yang nilai *recall* nya 0 artinya pada data tersebut data dengan target 1 tidak dapat diklasifikasikan dengan benar, atau semua data dikenali sebagai data yang targetnya 0.

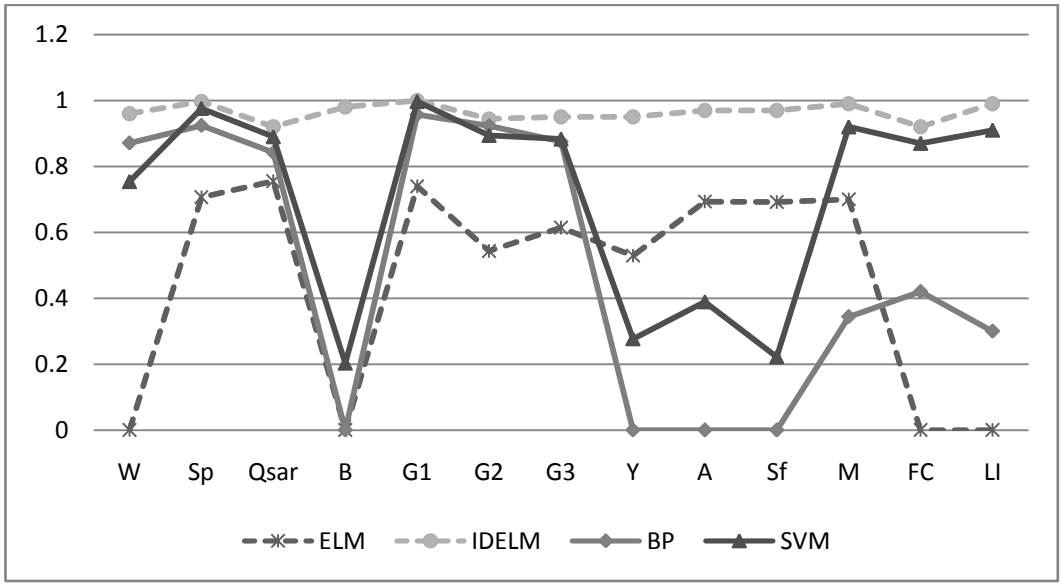
d. *G-mean*

Hasil perhitungan *G-mean* Integrasi seleksi data dan klasifikasi pada ELM (IDELM) dibandingkan dengan ELM, Backpropagation (BP), dan SVM menggunakan *benchmark data* ditunjukkan pada Tabel 4.13.

Tabel 4.13 *G-mean* dan *Specificity* dari uji menggunakan *benchmark data*

Data	***Specificity***				***G-mean***			
	ELM	**IDELM**	**BP**	**SVM**	**ELM**	**IDELM**	**BP**	**SVM**
W	1	0.92	0.999	0.991	0	0.96	0.871	0.754
Sp	0.633	0.993	0.945	0.988	0.706	0.997	0.924	0.976
Qsar	0.907	0.85	0.918	0.939	0.755	0.92	0.842	0.890
B	0.925	0.95	1	0.522	0	0.98	0	0.203
G1	0.734	1	0.93	0.993	0.739	1	0.957	0.997
G2	0.492	0.9	0.9	0.942	0.543	0.944	0.923	0.894
G3	0.514	0.91	0.984	0.9	0.614	0.95	0.875	0.883
Y	0.701	0.91	0.998	0.801	0.529	0.95	0	0.277
A	0.601	0.93	0.999	0.794	0.693	0.97	0	0.389
Sf	0.634	0.94	1	0.589	0.692	0.97	0	0.222
M	0.7	0.98	0.89	0.88	0.7	0.99	0.344	0.92
FC	1	0.85	0.99	0.82	0	0.92	0.42	0.87
LI	1	0.98	0.99	0.9	0	0.99	0.3	0.91
Rata-rata	0.7570	0.9317	0.9725	0.8583	0.4593	0.9649	0.4981	0.7065

G-mean(*Geometric mean*) merupakan salah satu alat ukur performa untuk evaluasi data yang mempunyai karakteristik *imbalance. G-mean* biasanya digunakan untuk mengukur akurasi dari suatu data *imbalanced* . Dari Tabel 4.13 dapat dilihat bahwa *G-mean* rata-rata dari IDELM paling tinggi dibandingkan ELM, BP, dan SVM. Artinya akurasi *imbalanced data* yang terbaik dari data yang digunakan di atas adalah pada algoritma IDELM yaitu 96.49%, sedangkan ELM, BP, dan SVM berturut – turut adalah 45.93%, 49.81%, dan 70.65%. Untuk lebih jelas melihat perbandingan *G-mean* setiap data, dapat dilihat Gambar 4.11.



Gambar 4.11 Grafik *G-mean* dari uji menggunakan *benchmark data*

Sama halnya dengan *precision* dan *recall*, untuk *benmarck data*, IDLM mempunyai nilai *G-mean* yang hampir selalu berada di atas 80%, sedangkan ELM, BP dan SVM tergantung pada jumlah data, banyak atribut yang digunakan, dan perbandingan data *imbalanced* nya.

e. *CPU Time*

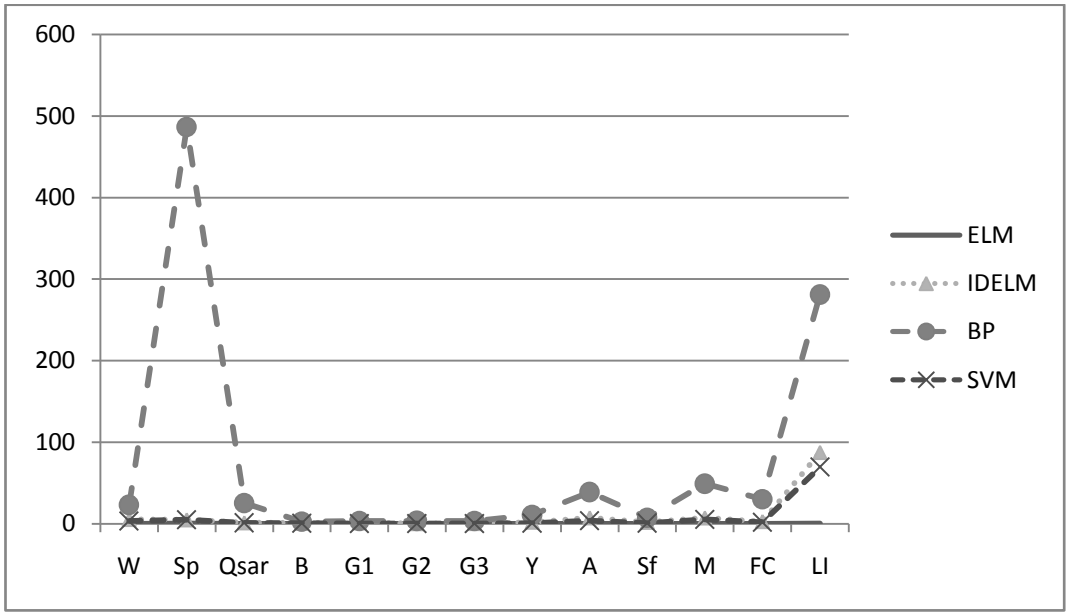
CPU Time yang dimaksud disini adalah waktu yang dibutuhkan untuk *training* setiap data. *CPU Time* yang dibutuhkan setiap data dan setiap algoritma untuk *training* data ditunjukkan pada Tabel 4.14.

Tabel 4.14 CPU Time ujicoba dengan *benchmark data*

Data	***CPU Time***			
	ELM(s)	**IDELM(s)**	**BP(s)**	**SVM(s)**
W	0.0281	5.2947	23.2067	3.2448
Sp	0.2714	4.8298	486.2551	4.8634
Qsar	0.0406	1.198	25.3065	1.10767
B	0.0187	0.911	2.6458	0.4930
G1	0.0094	1.198	3.4195	0.1997
G2	0.0094	0.2028	3.6099	0.1966
G3	0.0218	0.3762	3.2823	0.1903
Y	0.0218	1.95	10.8889	0.7363
A	0.0281	7.279	38.8973	3.6878
Sf	0.0125	1.797	7.4038	0.7582
M	0.109	7.179	49.1247	5.257
FC	0.078	2.719	30.1706	1.7784
LI	0.1903	87.2435	280.9079	69.4828
Rata-rata	**0.0591**	**9.3229**	**74.2399**	**7.0766**

Dari Tabel 4.14 dapat dilihat bahwa ELM adalah algoritma yang membutuhkan *CPU Time* yang sangat cepat. Dari data di atas rata-rata *CPU Time* yang sangat cepat adalah ELM yaitu 0.0591 detik, sedangkan *CPU Time* yang paling banyak adalah pada Backpropagation, yaitu 74.2399 detik. Sedangkan IDELM dan SVM berturut-turut adalah 9.3229 dan 7.0766 detik.

Untuk lebih detail melihat perbandingan *CPU Time* setiap data dapat dilihat pada Gambar 4.12.



Gambar 4.12 Grafik CPU *time* ujicoba dengan *benchmark data*

Dari Gambar 4.12 dapat dilihat bahwa waktu yang dibutuhkan untuk *training* data IDELM dan SVM selisih rata-rata hampir dua detik sedangkan BP membutuhkkan waktu yang cukup lama untuk proses *training* terutama pada data Spambase dan Letter Img, pada data spambase sangat tinggi grafiknya karena pada data tersebut menggunakan 51 atribut sedangkan pada data Letter Img merupakan data yang besar yaitu 20.000 akan tetapi atribut yang digunakan tidak sebanyak pada data Spambase. Sehingga waktu yang dibutuhkan untuk *training* data Spambase dan Letter Img jauh lebih banyak Spambase.

4.4.2 Prediksi *Binding site* Protein-Ligan dengan Integrasi Seleksi Data dan *Extreme Learning Machine*

Dalam penelitian ini, data yang digunakan merupakan data data protein hasil eksperimen yang dipublish dalam web RCSB Protein data bank, yang juga merupakan data open source. Data protein dengan *binding site* nya adalah data yang mempunyai karakter *imbalance*, sehingga untuk mengkur performa algoritma yang digunakan untuk klasifikasi tidak dapat menggunakan perhitungan akurasi biasa. Dalam penelitian ini ukuran performa dilihat dari *precision*, *recall*, *specificity*, dan *G-mean*. Dalam penilitian ini menggunakan

17 data protein. Dalam penelitian ini proses *training* dan *testing* adalah digunakan 1 protein untuk *testing* dan 1 protein sisa protein yang lain yang sejenis digunakan untuk *training*. Sehingga proses *training* dilakukan sebanyak data protein sisa dari protein yang sejenis.

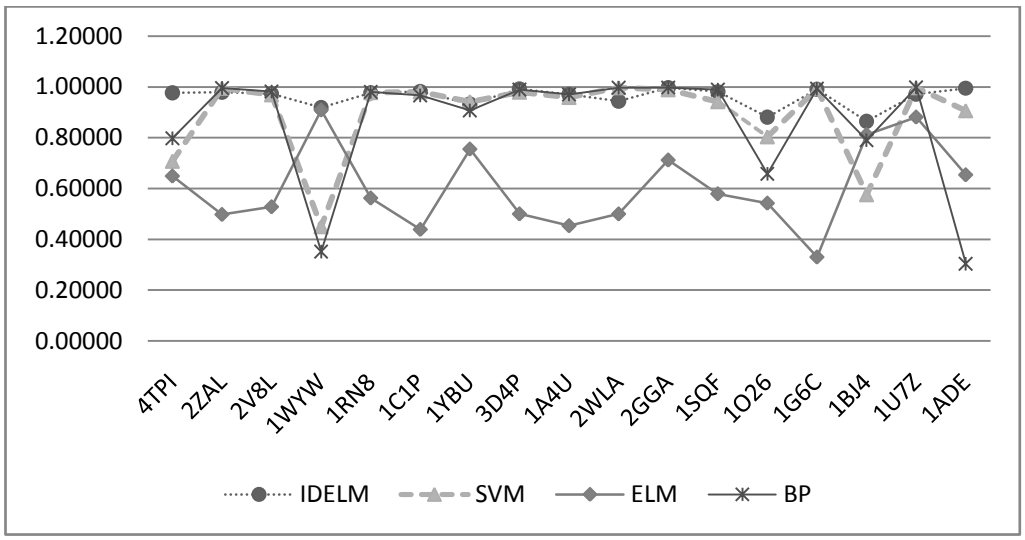
a. Akurasi

Hasil perhitungan akurasi integrasi seleksi data dan klasifikasi pada ELM, ELM, Backpropagation, dan SVM untuk prediksi *binding site* protein-ligan ditunjukkan pada Tabel 4.15.

Tabel 4.15 Akurasi prediksi *binding site* protein-ligan

Protein ID	**Akurasi**			
	IDELM	**SVM**	**BP**	**ELM**
4TPI	0.97661	0.7073	0.7977	0.6490
2ZAL	0.97867	0.9949	0.9950	0.4979
2V8L	0.97265	0.9690	0.9806	0.5280
1WYW	0.91824	0.4497	0.3516	0.9085
1RN8	0.97656	0.9777	0.9796	0.5625
1C1P	0.98106	0.9809	0.9661	0.4391
1YBU	0.93202	0.9422	0.9068	0.7551
3D4P	0.99224	0.9794	0.9889	0.5000
1A4U	0.96987	0.9586	0.9695	0.4536
2WLA	0.94333	0.9970	0.9963	0.5000
2GGA	0.99674	0.9890	0.9967	0.7119
1SQF	0.98093	0.9420	0.9885	0.5791
1O26	0.88080	0.8036	0.6581	0.5424
1G6C	0.99012	0.9930	0.9925	0.3304
1BJ4	0.86379	0.5756	0.7897	0.8119
1U7Z	0.97152	0.9971	0.9976	0.8810
1ADE	0.99482	0.9063	0.3040	0.6538
Rata-rata	0.96000	0.891959	0.862306	0.606129

Akurasi adalah porsi dari data yang diprediksi dengan benar oleh algoritma. Dari Tabel 4.15 dapat kita lihat bahwa rata – rata akurasi untuk prediksi *binding site* protein-ligan dengan menggunakan integrasi seleksi data dan ELM mempunyai akurasi rata-rata lebih tinggi dibandingkan dengan ELM, BP, dan SVM yaitu 0.96000. Artinya akurasi prediksi *binding site* protein-ligan dengan integrasi seleksi data dan ELM adalah 96%. Untuk lebih jelas perbandingan tiap data, dapat dilhat pada Gambar 4.13.



Gambar 4.13 Akurasi prediksi *binding site* protein-ligan

Dari grafik di atas terlihat bahwa akurasi IDELM dan BP hampir sama, semuanya di atas 80% di semua data. Kecuali pada data 1WYW, 1O26 dan 1ADE, BP mengalami penurunan yang cukup signifikan, hal tersebut dikarenakan data *training* lebih kecil di bandingkan data *testing*. Hal ini berbeda pada saat uji coba dengan *benchmark data* yang mempunyai akurasi yang berbeda-beda dan hampir semuanya di atas 80%. Persentase akurasi ini tidak dapat mengetahui apakah yang terklasifikasikan dengan benar adalah yang mempunyai target 1 (*binding site)* atau yang target 0. Sehingga, akurasi biasa seperti perhitungan ini tidak dapat menggambarkan performa algoritma yang digunakan, terutama pada masalah ini, karena data yang digunakan dalam penelitian ini adalah data yang *imbalanced* .

b. Recall

Hasil perhitungan *recall* integrasi seleksi data dan klasifikasi pada ELM, ELM, Backpropagation, dan SVM untuk prediksi *binding site* protein-ligan ditunjukkan pada tabel 4.16

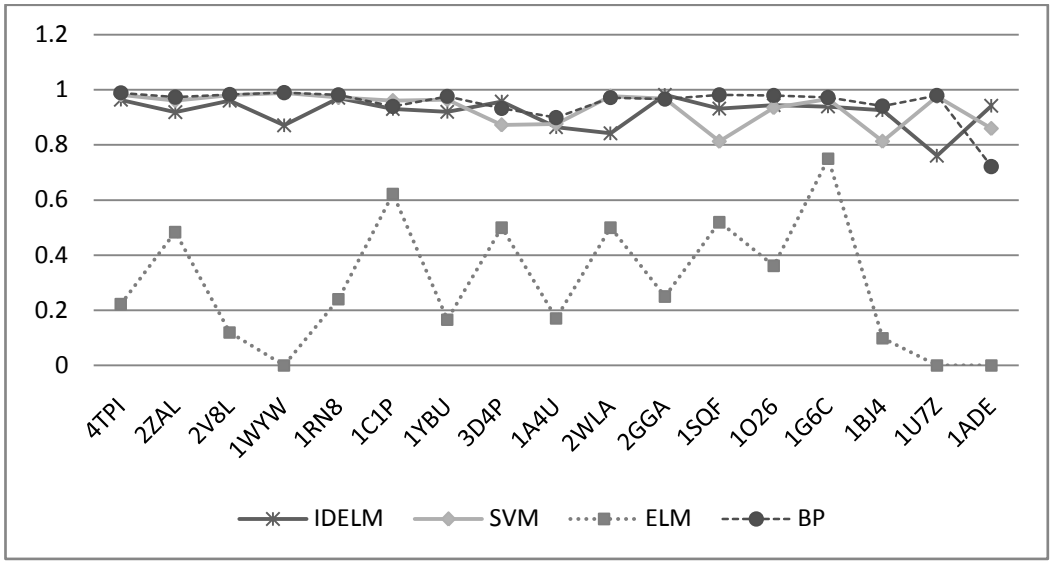
Tabel 4.16 *Recall* prediksi *binding site* protein-ligan

Protein ID	***Recall***			
	IDELM	**SVM**	**BP**	**ELM**
4TPI	0.962844	0.98174	0.98174	0.22208
2ZAL	0.917986	0.96076	0.97287	0.48377
2V8L	0.959871	0.98026	0.98301	0.11958
1WYW	0.870415	0.98864	0.9897	0
1RN8	0.969862	0.97222	0.98148	0.24056
1C1P	0.930024	0.96076	0.93924	0.62199
1YBU	0.91922	0.96425	0.97632	0.1662
3D4P	0.956885	0.87247	0.93219	0.5
1A4U	0.863636	0.87585	0.89892	0.17096
2WLA	0.842199	0.97674	0.97166	0.5
2GGA	0.981013	0.96756	0.96598	0.25
1SQF	0.930942	0.81263	0.98144	0.51963
1O26	0.943318	0.93525	0.97954	0.36149
1G6C	0.938536	0.96616	0.97238	0.75
1BJ4	0.925052	0.81289	0.94078	0.09853
1U7Z	0.760602	0.97538	0.97948	0
1ADE	0.941615	0.85934	0.72104	0
Rata-rata	0.918472	0.93311	0.95105	0.2944

Recall adalah porsi dari data sampel yang diprediksi benar oleh algoritma. Dari Tabel 4.16 dapat kita lihat bahwa integrasi seleksi data dan klasifikasi dapat jauh meningkatkan nilai *recall* dibandingkan ELM biasa, dalam kasus ini adalah pada masalah data *imbalanced* . Rata- rata *recall* untuk integrasi seleksi data dan ELM adalah 0.918472 sedangkan nilai rata-rata *recall* pada ELM biasa untuk data protein di atas adalah 0.2944 saja, artinya dari semua data *binding site* yang digunakan, sebanyak 91.8472% data dapat dikenali benar oleh

IDELM. Rata-rata *recall* dari SVM dan BP lebih baik dibandingkan dengan IDELM yaitu 93.311% dan 95.105%.

Untuk dapat melihat lebih jelas perbandingan nilai *recall* antara ELM,IDELM,BP dan SVM untuk prediksi *binding site* protein ligan dapat dilihat pada Gambar 4.14.



Gambar 4.14 R*ecall* prediksi *binding site* protein-ligan

Dari grafik di atas dapat kita lihat bahwa nilai *recall* hampir pada setiap data untuk IDELM, BP, dan SVM selalu berada di atas 80%, sedangkan ELM selalu jauh berada di bawah dibandingkan ketiga algoritma yang lain. ELM memiliki nilai *recall* yang sangat rendah hal ini dikarenakan data yang digunakan merupakan *imbalanced data*. Sedangkan SVM dan BP meskipun data yang digunakan merupakan data *imbalance* karena attribut yang digunakan tidak banyak maka BP dan ELM tetap bisa memperoleh nilai *recall* yang tinggi.

c. G-mean

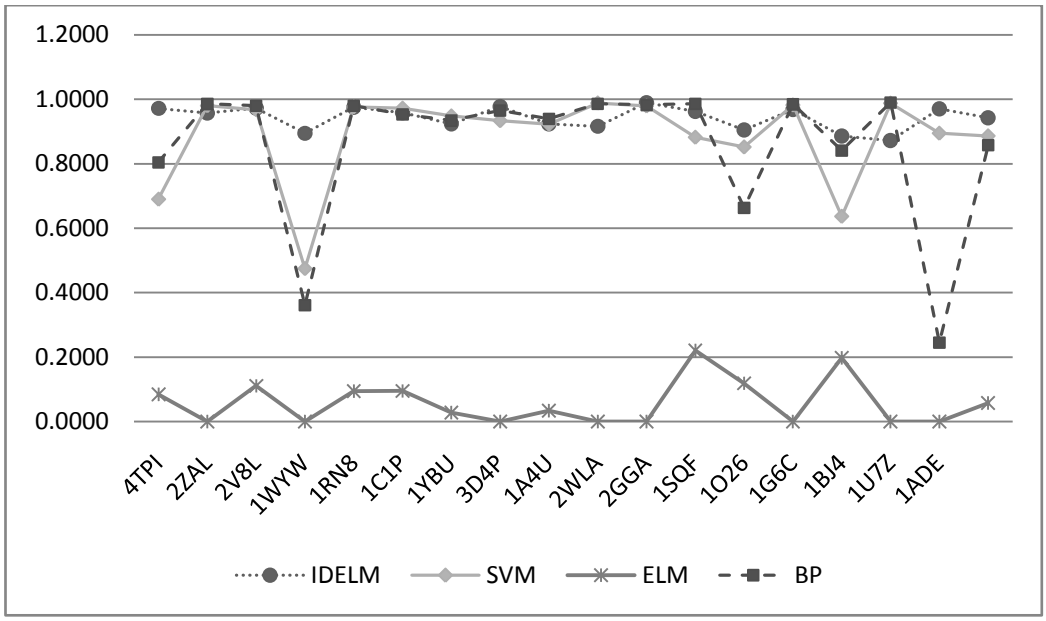
Perhitungan *specificity* dan *G-mean* integrasi seleksi data dan klasifikasi pada ELM, ELM, *Backpropagation*, dan SVM untuk prediksi *binding site* protein-ligan.

Tabel 4.17 S*pecificity* dan *G-mean* prediksi *binding site* protein-ligan

Protein ID	***Specificity***				***G-mean***			
	IDELM	**SVM**	**BP**	**ELM**	**IDELM**	**SVM**	**BP**	**ELM**
4TPI	0.9813	0.6133	0.7323	0.7952	0.9720	0.77593	0.84788	0.42024
2ZAL	0.9990	1.0000	0.9983	0.5000	0.9571	0.98018	0.98548	0.49182
2V8L	0.9854	0.9578	0.9782	0.9364	0.9723	0.96895	0.98058	0.33463
1WYW	0.9227	0.3991	0.2917	1.0000	0.8945	0.62814	0.53731	0
1RN8	0.9812	0.9815	0.9783	0.7873	0.9754	0.9765	0.9798	0.0944
1C1P	0.9899	0.9844	0.9707	0.4076	0.9588	0.97249	0.95485	0.5035
1YBU	0.9338	0.9391	0.8971	0.8365	0.9235	0.95159	0.93589	0.37287
3D4P	0.9989	0.9996	0.9996	0.5000	0.9777	0.93388	0.96531	0.5
1A4U	0.9898	0.9741	0.9828	0.5066	0.9238	0.92369	0.93993	0.2943
2WLA	0.9969	1.0000	1.0000	0.5000	0.9163	0.9883	0.98573	0.5
2GGA	0.9980	0.9907	0.9992	0.7500	0.9894	0.97908	0.98246	0.43301
1SQF	0.9945	0.9772	0.9905	0.5953	0.9622	0.89111	0.98595	0.55616
1O26	0.8703	0.7815	0.6042	0.5727	0.9048	0.85493	0.76928	0.45501
1G6C	1.0000	0.9981	0.9963	0.2500	0.9687	0.98202	0.98426	0.43301
1BJ4	0.8560	0.5452	0.7704	0.9032	0.8858	0.66572	0.85133	0.29831
1U7Z	1.0000	1.0000	1.0000	1.0000	0.8721	0.98761	0.98969	0
1ADE	1.0000	0.9312	0.0832	1.0000	0.9704	0.89453	0.24497	0
Rata-rata	0.9705	0.8866	0.8396	0.6965	0.9426	0.9032	0.8777	0.3546

G-mean(*Geometric mean*) merupakan salah satu alat ukur performa untuk evaluasi data yang mempunyai karakteristik *imbalance*. Untuk menghitung *G-mean* terlebih dahulu *sensitifity* dan *specificity*. *Sensifity* adalah untuk mengukur akurasi sampel positif, untuk menghitung *sensitifity* sama dengan perhitungan *recall*. Sedangkan *specificity,* adalah akurasi untuk sampel negatif. Dari tabel di atas dapat dilihat bahwa *G-mean* rata-rata dari ELM, IDELM, BP, dan SVM dari 17 data yang digunakan berturut-turut adalah 0.3546, 0.9426, 0.8777, dan 0.9032.

Untuk lebih terlihat perbandingan nilai *G-mean* pada setiap data yang digunakan dapat dilihat pada Gambar 4.15.



Gambar 4.15 *G-mean* prediksi *binding site* protein-ligan

Dari Gambar 4.15 dapat dilihat bahwa nilai *G-mean* IDELM bagus untuk prediksi *binding site* protein-ligan dengan dua atribut seperti yang digunakan dalam penelitian ini. Karena dari keempat algoritma tersebut nilai *G-mean* IDELM paling tinggi, dan rata-rata pada setiap data selalu di atas 80%.

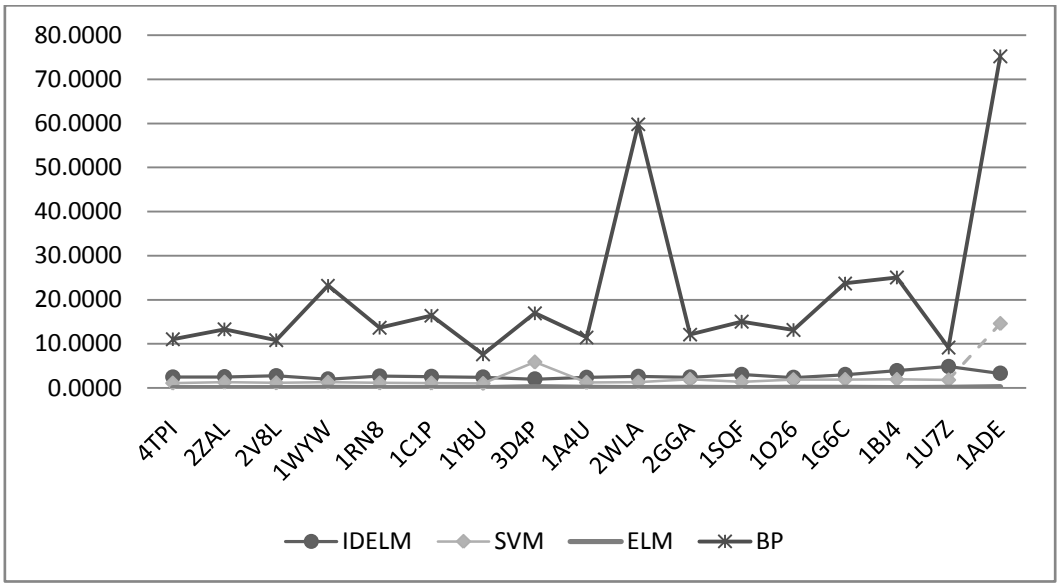
d. *CPU Time*

CPU Time yang dimaksud disini adalah waktu yang dibutuhkan untuk *training* setiap data. *CPU Time* yang dibutuhkan setiap data dan setiap algoritma untuk *training* data ditunjukkan pada Tabel 4.18.

Dari Tabel 4.18 dapat dilihat bahwa ELM adalah algoritma yang memiliki *CPU Time* yang sangat cepat. Dari tabel 4.18 dapat dilihat rata-rata *CPU Time* yang paling cepat adalah ELM yaitu 0.069283 detik, sedangkan *CPU Time* yang paling banyak adalah pada BP, yaitu 22.18663 detik. Sedangkan IDELM dan SVM berturut-turut adalah 2.786071 dan 2.4443 detik. Untuk lebih detail melihat perbandingan *CPU Time* setiap data dapat dilihat pada Gambar 4.16.

Tabel 4.12 CPU time data *binding site* protein-ligan

Protein ID	*CPU Time (Training)*				*CPU Time (Testing)*			
	IDELM (s)	**SVM (s)**	**BP (s)**	**ELM (s)**	**IDELM (s)**	**SVM (s)**	**BP (s)**	**ELM (s)**
4TPI	2.4752	1.0764	11.0189	0.0390	0.0416	0.1326	0.1482	0.0026
2ZAL	2.5064	1.2766	13.2679	0.0442	0.0442	0.0442	0.1742	0.0052
2V8L	2.7742	1.1128	10.7979	0.0234	0.0260	0.1170	0.1300	0.0130
1WYW	1.9864	1.2610	23.1583	0.0702	0.1222	0.0754	0.2132	0.0052
1RN8	2.6910	1.1050	13.6319	0.0156	0.0182	0.0338	0.1378	0.0026
1C1P	2.5818	1.0816	16.3515	0.0364	0.0598	0.0546	0.1690	0.0078
1YBU	2.4310	0.9906	7.5920	0.0416	0.0728	0.1664	0.1846	0.0078
3D4P	1.9890	5.8656	16.9183	0.2262	0.0468	0.0936	0.1872	0.0078
1A4U	2.3946	1.2090	11.4115	0.0624	0.0546	0.0676	0.1560	0.0000
2WLA	2.6130	1.2948	59.7796	0.0468	0.0312	0.0364	0.1716	0.0078
2GGA	2.4414	1.9071	12.0589	0.0624	0.0741	0.1755	0.1599	0.0117
1SQF	3.0498	1.3416	15.0021	0.0416	0.0507	0.0546	0.1924	0.0052
1O26	2.3634	1.8447	13.1353	0.1014	0.1131	0.0520	0.2106	0.0039
1G6C	3.0147	1.8876	23.6654	0.0546	0.0702	0.0585	0.1833	0.0078
1BJ4	3.9078	1.9032	25.0187	0.0468	0.0546	0.0312	0.1521	0.0039
1U7Z	3.8516	1.7940	9.1417	0.0624	0.1092	0.0832	0.1716	0.0000
1ADE	3.2916	14.6014	95.2230	0.2028	0.0936	0.0546	0.1404	0.0156
Rata-rata	2.7861	2.4443	22.187	0.0693	0.0637	0.0784	0.1696	0.0063



Gambar 4.16 Grafik CPU time data *binding site* protein-ligan

Dari Gambar 4.16 dapat dilihat bahwa CPU Time pada data ini tergantung pada banyak data, karena data dalam kasus ini semuanya memiliki dua atribut. Dari grafik di atas juga sangat terlihat bahwa waktu komputasi ELM jauh lebih cepat disbanding BP, SVM, dan IDELM.

4.5 Pembahasan

Dalam penelitian ini terdapat dua hal yang dilakukan, yaitu melakukan integrasi seleksi data dan *Extreme Learning Machine* (IDELM) dan prediksi *binding site* protein-ligan dengan menggunakan IDELM. Integrasi seleksi data dan *Extreme Learning Machine* (IDELM) terlebih dahulu dilakukan pengujian performa dibandingkan dengan ELM, BP, dan SVM. dalam penelitian ini pengukuran performa dilakukan dengan pengujian menggunakan *Benchmark data* yang diambil dari *UCI Machine Learning dan Benchmark data* dan *Benchmark Data Sets for Highly Imbalanced Binary Classification.*

Dalam pengujian dengan menggunakan *benchmark data* pembagian data *training* dan data *testing* menggunakan 5-*fold cross validation*. Arti dari 5-*fold cross validation* adalah, dari seluruh data yang ada dibagi menjadi 5 kelompok data, 1 kelompok data untuk *testing* 4 kelompok data yang lain digunakan untuk *training*. *Training* dilakukan sebanyak lima kali dengan data *testing* yang berbeda, sehingga setiap data pernah digunakan untuk *training* maupun *testing*. *Akurasi, precision, recall*, *specificity* dan *G-mean* diambil dari rata-rata lima kali *training*. Selain digunakan untuk membagi kelompok data *training* dan *testing k-fold cross validation* juga berfungsi sebagai validasi dari metode yang dipakai.

Dengan pengujian dengan mengunakan *Benchmark data* IDELM mempunyai performa yang lebih bagus dibandingkan Backpropagation, SVM, dan ELM. Hasil akurasi IDELM lebih bagus 0.38% dibandingakn BP, sedangkan akurasi IDELM 11.61% lebih baik jika dibandingkan dengan SVM dan lebih baik 15.76% dibandingkan ELM untuk 13 *benchmark data* yang digunakan. Ukuran akurasi biasa pada masalah ini tidak dapat dijadikan sebagai ukuran performa suatu metode yang digunakan, karena *Benchmark data* yang digunakan dalam penelitian ini mempunyai karakter *imbalanced* data. Hal tersebut dianalogikan

pada masalah data yang misalnya mempunyai 98 data negatif dan 2 data positif, maka meskipun algoritma tidak dapat mengklasifikasikan data positif akurasi yang diperoleh masih 98%. Maka dari itu, akurasi kurang cocok jika digunakan untuk mengukur performa suatu algoritma untuk kasus data yang *imbalanced* .

Pada pengukuran *precision* yaitu akurasi dari data yang diklasifikasikan menjadi data positif. IDELM tetap mempunyai persentase yang paling tinggi dibandingkan ELM, SVM, dan BP masing- masing lebih baik 58.08 %, 57.31%, dan 36.54%. Pada pengukuran *recall*, yaitu akurasi data positif, IDELM tetap mempunyai persentase yang paling tinggi. Akan tetapi pada pengukuran *recall* SVM mempunyai persentase yang lebih tinggi dibandingkan Backpropagation. Hal ini menunjukkan meskipun akurasi dari BP lebih baik dibandingkan SVM akan tetapi *recall* SVM lebih baik dibandingkan BP. Artinya, jika dilihat dari keseluruhan data BP dapat mengklasifikasikan data dengan benar lebih banyak dibandingkan SVM, tanpa melihat data positif atau negatif. Jika akurasi dilihat hanya pada akurasi data positif saja, SVM dapat mengklasifikasikan data positif dengan benar lebih banyak dibandingkan BP.

Pada pengukuran s*pecificity*, yaitu akurasi data negatif, BP mempunyai persentase terbaik dibandingkan ELM, IDELM, dan SVM. Artinya jika dilihat dari pengklasifikasian data negatif, BP dapat mengklasifikasikan data negatif dengan benar lebih banyak dibandingkan ELM, IDELM dan SVM. Dengan persentase selisih BP dengan ELM, IDELM dan SVM masing- masing adalah 21.55%, 4.08%, dan 11.42%.

G-mean merupakan ukuran performa yang sering digunakan untuk mengukur performa dari suatu metode pada kasus *imbalanced* data. Dalam penelitian ini *G-mean* juga digunakan untuk mengukur performa IDELM, ELM, BP, dan SVM. *G-mean* mengindikasikan keseimbangan antara kinerja klasifikasi pada kelas mayoritas dan minoritas. Dari hasil *G-mean* pada sub bab sebelumnya dapat dilihat bahwa performa IDELM lebih baik dibandingkan ELM, BP, dan SVM untuk kasus *imbalanced* data dengan 13 *benchmark data* yang digunakan.

Karena pada saat uji dengan menggunakan *benchmark data* IDELM sudah diketahui performanya untuk masalah *imbalanced data*, selanjutnya IDELM diterapkan untuk prediksi *binding site* protein-ligan. IDELM di uji

dengan menggunakan data *imbalanced* data karena data *binding site* protein ligan juga mempunyai sifat *imbalanced* , artinya data yang berpotensi sebagai binding site jauh lebih sedikit dibandingkan dengan data yang tidak berpotensi sebagai binding site. Dari data protein yang sudah diperoleh, protein dikelompokkan berdasarkan jenisnya. Pembagian data *training* dan *testing* untuk prediksi binding site adalah 1 protein untuk *training* dan 1 protein untuk *testing* dilakukan *training* sebanyak data protein pada setiap jenis protein.

Untuk kasus prediksi *binding site* protein ligan dengan 17 data yang digunakan IDELM memiliki performa yang lebih baik dibandingkan dengan ELM, BP, dan SVM. Meskipun mempunyai performa yang bagus, tetapi pada beberapa data seperti 1WYW, 1O26 dan 1ADE IDELM mengalami penurunan performa, hal tersebut dikarenakan banyak data protein untuk *training* lebih sedikit dibandingkan data protein untuk *testing*, selain itu banyak atribut juga mempengaruhi performa dari IDELM. Selain jumlah atribut dan lebih sedikitnya data *training* dibandingkan data *testing*, performa IDELM juga dipengaruhi banyaknya *hidden node*. Sedangkan BP dan SVM juga sama, pada ketiga data terssebut performa BP dan SVM turun cukup signifikan, hal tersebut juga dikarenakan banyak data *training* jauh lebih sedikit dibandingkan data *testing* dan perbandingan data positif dan negatif yang jauh.

Dalam kasus *imbalanced data*, integrasi seleksi data dan ELM dapat meningkatkan performa dari ELM biasa. Dari uraian di atas dapat dibuat dilihat bahwa IDELM mempunyai performa lebih baik dibandingkan ELM, BP, dan SVM jika dilihat dari *G-mean* atau ukuran akurasi untuk imbalanced data. Jika dilihat dari performa waktu komputasi, ELM mempunyai waktu komputasi yang sangat cepat dibandingkan dengan IDELM, SVM, dan BP.

BAB 5
KESIMPULAN DAN SARAN

Pada bab ini disampaikan kesimpulan dan saran yang diperoleh dari hasil integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* protein ligan.

5.1 Kesimpulan

Berdasarkan dari uraian dan pembahasan bab sebelumnya hasil dari penelitian serta menjawab dari rumusan masalah prediksi *binding site* protein ligan dengan integrasi seleksi data dan ELM adalah sebagai berikut:

a. Integrasi seleksi data dan ELM dalam penelitian ini dilakukan dengan cara menggabungkan proses seleksi data dalam proses *training* pada ELM. Pengintegrasian seleksi data dan ELM pada proses *training* memanfaatkan MSE (*Mean Square Error*) dan *input* data secara *sequential*. Pengujian MSE dilakukan untuk proses seleksi data setiap kelompok data, dengan ketentuan jika $\Delta MSE \geq 0$ maka kelompok data tersebut digunakan sebagai data *training*, selanjutnya dilakukan *update* bobot.
b. Dari penelitian dengan uji *benchmark data* ini dari 13 data yang digunakan menghasilkan *G-mean* rata-rata adalah sebesar 96.49%. *CPU Time* yang dibutuhkan untuk *training* setiap data rata- rata adalah 9.3229 detik.
c. Dari penelitian untuk prediksi *binding site* protein-ligan dari 17 data protein yang digunakan menghasilkan *G-mean* rata-rata adalah sebesar 94,26%. *CPU Time* yang dibutuhkan untuk *training* 1 data protein rata- rata adalah 2.78607 detik.
d. Untuk prediksi *binding site* protein-ligan integrasi seleksi data dan ELM mempunyai performa paling baik dibandingkan ELM biasa, BP, dan SVM, karena nilai *G-mean* integrasi seleksi data dan ELM lebih tinggi dibandingkan ELM biasa, BP, dan SVM.

5.2 Saran

Penelitian ini masih perlu banyak lagi perbaikan untuk meningkatkan performa prediksi *binding site* protein ligan, beberapa saran dari penulis untuk pembaca dan peneliti adalah sebagai berikut:

a. Untuk bisa diaplikasikan dalam desain obat atau yang lain(vaksinasi, pengembangan bibit unggul, maupun vitamin), perlu ada pengkajian lebih mendalam dari segi kimia dan biologi,
b. Untuk penerapan IDELM dalam prediksi *binding site* protein-ligan maupun protein-protein, sebaiknya data *training* dipilih protein yang sejenis yang banyak datanya lebih dari protein yang diprediksi.

DAFTAR PUSTAKA

An, J., Totrov, M. dan Abagyan, R., (2005), "Pocketome via Comprehensive Identification and Classification of Ligan Binding Envelopes", *Molecular & Cellular Proteomics,* Vol. 4, hal. 752-761.

Batuwitage, M.R.K., (2010), "Enhanced Class Imbalance Learning Methods for Support Vector Machines", *Thesis of Doctor of Philosophy Hilary Term 2010,* St. Cross College.

Bekkar, M., dan Alitouche, T.A., (2013), "Imbalanced Data Learning Approaches Review", *International Journal of Data Mining & Knowledge Management Process (IJDKP),* Vol. 4, No. 4, hal. 15-33

Capra, J.A. dan Singh, M., (2007), "Predicting functionally important residues from *sequence* conservation", *Sequence analysis,* Vol. 23, no. 15, hal. 1875–1882

Chawla, N.V., (2010), "Data Mining for Imbalanced Datasets: An Overview", Data Mining and Knowledge Discovery Handbook, Springer, New York.

Ferrari, S dan Stengel, R.F., (2005) ," Smooth function approximation usingneural networks", *IEEE Trans. Neural Networks,* Vol. 16, No. 1, hal 24–38.

Ganganwar, V., (2012), "On overview of classification algorithms for *imbalanced* datasets", *International Journal of Emerging Technology and Advanced Engineering*. Vol 2, no 4, hal. 42-47.

Gutteridge, A., Bartlett, G., dan Thornton, J., (2003), "Using a Neural Network and Spatial Clustering to Predict the Location of Active Sites in Enzymes,", *J. Molecular Biology,* vol. 330, hal. 719-734.

He, H., Garcia, E.A., (2009), "Learning from *imbalanced* data". *IEEE Trans. Knowl. Data Eng,* Vol. 21, no.9, hal. 1263-1284.

Hendlich, M., Rippmann, F. dan Barnickel, G., (1997), "LIGSITE: Automatic and Efficient Detection of Potential Small Molecule-*Binding Site*s in Proteins", *Journal of Molecular Graphics and Modelling*, Vol.15, hal. 359 –363

Huang, G., Zhu, Q. dan Siew, C., (2006a), "*Extreme Learning Machine*: Theory and Applications", *Neurocomputing*, Vol. 70, 489–501

Huang, G., Chen, L. dan Siew, C., (2006b), "Universal Approximation Using Incremental Constructive Feedforward Networks With Random *Hidden node*s", *IEEE Transactions on Neural Networks*, vol. 17, no. 15, hal 879-892.

Hulse, J.V., Khoshgoftaar, T.M., dan Napolitano, A., (2007), "Experimental Perspectives on Learning from Imbalanced Data", *Proceedings of the 24th International Conference on Machine Learning*.

Imah, E.M. , Jatmiko, W. , dan Basarudin, T., (2012), "Adaptive Multilayer Generalized Learning Vector Quantization (AMGLVQ) as new algorithm with integrating feature extraction and classification for Arrhythmia heartbeats classification", *IEEE International Conference Systems, Man, and Cybernetics (SMC)*, hal. 150-155.

Keil, M., Exner, T., dan Brickmann, J., (2004),"Pattern Recognition Strategies for Molecular Surfaces: *Binding site* Prediction with a Neural Network," *Journal Computational Chemistry*, vol. 25, no. 6, hal. 779-789.

Koike, A. and Takagi, T., (2004) "Prediction of Protein-Protein Interaction Sites Using Support Vector Machines", *Protein Engginering Design and Selection*, vol. 17, no. 2, hal. 165-173.

Laskowski, Roman, A., (1995), "SURFNET: A Program for Visualizing Molecular Surfaces, Cavities, and Intermolecular Interactions", *Journal of Molecular Graphics*, Vol. 13, hal. 323-330

Leckband, Deborah, (2000), "Measuring The Forces That Control Protein Interaction", *Annu. Rev. Biophys. Biomol. Struct*, Vol. 29, hal 1-26.

Liang, N., Huang, G., Saratchandran, P., dan Sundararajan, N., (2006), "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks", *IEEE Transactions on Neural Networks,*Vol. 17, No. 6, hal 1411-1424.

Lopez, G., Valencia, A, dan Tress, M., (2007), "FireDB–a database of functionally important residues from proteins of known structure", *Nucleic Acids Research*, Vol. 35, hal 219-223.

Makridakis, S., Wheelwright, S.C., dan McGee, V.E, (1999), *Metode dan Aplikasi Peramalan*, Jakarta, Erlangga.

Macausland, R., (2014), "*The Moore-Penrose Inverse and Least Square*", University of Puget Sound.

Melisa, I., dan Oetama, R.S., (2013), Analisis Data Pembayaran Kredit Nasabah Bank Menggunakan Metode Data Mining, Ultima InfoSys, Vol IV, no.1. hal 18-27.

Mitchell, T.M.,(1997), *Machine Learning*, McGraw-Hill Science.

Motiejunas, D., dan Wade, R, (2006), "Structural, Energetics, and Dynamic Aspects of Ligan-Receptor Interactions", In J. B. Taylor & D. J. Triggle (Eds.), *Comprehensive Medicinal Chemistry II Volume 4: Computer-Assisted Drug Design*, Vol. 4, hal. 193-214. Elsevier.

Nguyen, G.H., Bouzerdoum, A., dan Phung, S.L., (2009), "Learning Pattern Classification Task with Imbalanced Data Sets", University of Wollongong, Australia.

Rovira, C.,(2005), "Study of Ligan–Protein Interactions by Means of Density Functional Theory and First-Principles Molecular Dynamics", dalam *Protein–Ligan Interactions Methods and Applications*, Vol 305,G. Ulrich Nienhaus, Humana Press, New Jersey, hal.517-553.

Schneider, G., dan Baringhaus, K.-H, (2008), *Molecular Design: Concepts and Applications*, WILEY-VCH.

Serre, D., (2002), *Matrices: Theory and Applications*, Springer, New York.

Shen, S. dan Tuszynski, J.A., (2008), *Theory and Mathematical Methods for Bioformatics*, Springer, Verlag Berlin Heidelberg.

Siang, J.J., (2009), *Jaringan Syaraf Tiruan dan Pemrogramanan Menggunakan MATLAB*, ANDI, Yogyakarta.

Trapsilasiwi dan Kemala, R., (2013), "Klasifikasi Multiclass Untuk *Imbalanced* Data Menggunakan SMOTE Least Square Support Vector Machine",Tesis Jurusan Statistika Institut Teknologi Sepuluh Nopember.

Valdar, W.S.J., (2002), "Scoring residue conservation", *Proteins*, Vol. 48, hal. 227–241.

Wang, D.D., Wang, R., dan Yan, H., (2014), "Fast Prediction of Protein–Protein Interaction Sites Based on *Extreme Learning Machine*s", *Neurocomputing*,Vol. 128, hal. 258–266.

Wang, D. dan Huang, G., (2005), "Protein *Sequence* Classification Using *Extreme Learning Machine*", *Proceedings of International Joint Conference on Neural Networks*, Montreal, Canada,hal. 1406-1411.

Widodo, U., Huswo, D, Ichsan, M.dan Putri, J.F., (2014), "Molecular Docking Workshop", Indonesian Institute of Bioinformatics, Malang.

Wong, G.Y., Leung, F.H.F. dan Ling, S.,(2012), "Predicting Protein-Ligan *Binding site* with Differential Evolution and Support Vector Machine", IEEE World Congress on Computational Intelligence, Brisbane, Australia

Wong, G.Y., Leung, F.H.F. dan Ling, S.H.,(2013), "Predicting Protein-Ligan *Binding site* Using Support Vector Machine with Protein Properties", Transactions on Computational Biology and Bioinformatics, Vol. 10, No. 10, hal. 1517-1529.

Zhu, C., Yin, J. dan Li, Q., (2013), "A Stock Decision Support System Based on ELM", *Proceedings of the International Conference on Extreme Learning Machines (ELM2013)*, (eds) Sun, F., Toh, K.-A., Romay, M.G., Mao, K., Beijing, hal.67-79.

Zvelebil, M. dan Baum, J.O., 2008, *Understanding Bioinformatics*, Garland Science, New York

Lampiran A
Contoh data Protein

```
HEADER    HYDROLASE                                21-DEC-04   1YBU
TITLE     MYCOBACTERIUM TUBERCULOSIS ADENYLYL CYCLASE RV1900C CHD, IN COMPLEX
TITLE    2 WITH A SUBSTRATE ANALOG.
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: LIPJ;
COMPND   3 CHAIN: A, B, C, D;
COMPND   4 FRAGMENT: RV1900C CHD;
COMPND   5 EC: 4.6.1.1;
COMPND   6 ENGINEERED: YES
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: MYCOBACTERIUM TUBERCULOSIS;
SOURCE   3 ORGANISM_TAXID: 83332;
SOURCE   4 STRAIN: H37RV;
SOURCE   5 GENE: RV1900C;
SOURCE   6 EXPRESSION_SYSTEM: ESCHERICHIA COLI;
SOURCE   7 EXPRESSION_SYSTEM_TAXID: 562;
SOURCE   8 EXPRESSION_SYSTEM_STRAIN: B834(DE3)[PREP4];
SOURCE   9 EXPRESSION_SYSTEM_VECTOR_TYPE: PLASMID;
SOURCE  10 EXPRESSION_SYSTEM_PLASMID: PQE30
KEYWDS    CYCLASE HOMOLOGY DOMAIN, CHD, RV1900C, HYDROLASE
EXPDTA    X-RAY DIFFRACTION
AUTHOR    S.C.SINHA,M.WETTERER,S.R.SPRANG,J.E.SCHULTZ,J.U.LINDER
REVDAT   4   13-JUL-11 1YBU    1       VERSN
REVDAT   3   24-FEB-09 1YBU    1       VERSN
REVDAT   2   15-MAR-05 1YBU    1       JRNL
REVDAT   1   15-FEB-05 1YBU    0
JRNL        AUTH   S.C.SINHA,M.WETTERER,S.R.SPRANG,J.E.SCHULTZ,J.U.LINDER
JRNL        TITL   ORIGIN OF ASYMMETRY IN ADENYLYL CYCLASES: STRUCTURES OF
JRNL        TITL 2 MYCOBACTERIUM TUBERCULOSIS RV1900C.
JRNL        REF    EMBO J.                       V.  24   663 2005
JRNL        REFN                   ISSN 0261-4189
JRNL        PMID   15678099
JRNL        DOI    10.1038/SJ.EMBOJ.7600573
REMARK   2
REMARK   2 RESOLUTION.    2.40 ANGSTROMS.
REMARK   3
REMARK   3 REFINEMENT.
REMARK   3   PROGRAM     : CNS 1.1
REMARK   3   AUTHORS     : BRUNGER,ADAMS,CLORE,DELANO,GROS,GROSSE-
REMARK   3               : KUNSTLEVE,JIANG,KUSZEWSKI,NILGES,PANNU,
REMARK   3               : READ,RICE,SIMONSON,WARREN
REMARK   3
REMARK   3  REFINEMENT TARGET : ENGH & HUBER
REMARK   3
REMARK   3  DATA USED IN REFINEMENT.
REMARK   3   RESOLUTION RANGE HIGH (ANGSTROMS) : 2.40
REMARK   3   RESOLUTION RANGE LOW  (ANGSTROMS) : 31.94
REMARK   3   DATA CUTOFF            (SIGMA(F)) : 0.000
```

```
REMARK   3   DATA CUTOFF HIGH         (ABS(F)) : 2109367.070
.
.
.

REMARK 800 SITE_DESCRIPTION: BINDING SITE FOR RESIDUE APC D 602
REMARK 900
REMARK 900 RELATED ENTRIES
REMARK 900 RELATED ID: 1YBT   RELATED DB: PDB
REMARK 900 SAME PROTEIN WITHOUT LIGAND
DBREF  1YBU A  291   462  UNP    O07732   O07732_MYCTU   291    462
DBREF  1YBU B  291   462  UNP    O07732   O07732_MYCTU   291    462
DBREF  1YBU C  291   462  UNP    O07732   O07732_MYCTU   291    462
DBREF  1YBU D  291   462  UNP    O07732   O07732_MYCTU   291    462
SEQADV 1YBU MET A  279  UNP  O07732             CLONING ARTIFACT
SEQADV 1YBU ARG A  280  UNP  O07732             CLONING ARTIFACT
.
.
.
.
SEQADV 1YBU HIS D  288  UNP  O07732             EXPRESSION TAG
SEQADV 1YBU GLY D  289  UNP  O07732             CLONING ARTIFACT
SEQADV 1YBU SER D  290  UNP  O07732             CLONING ARTIFACT
SEQRES   1 A  184  MET ARG GLY SER HIS HIS HIS HIS HIS HIS GLY SER ALA
SEQRES   2 A  184  GLU ARG MET LEU ALA THR ILE MET PHE THR ASP ILE VAL
.
.
.
SEQRES  14 D  184  TRP ARG LEU CYS VAL LEU MET ARG ASP ASP ALA THR ARG
SEQRES  15 D  184  THR ARG
FORMUL   5   MN    2(MN 2+)
FORMUL   7  APC    2(C11 H18 N5 O12 P3)
FORMUL   9  HOH   *223(H2 O)
HELIX    1   1 GLY A  305  GLY A  314  1                                  10
HELIX    2   2 GLY A  314  PHE A  336  1                                  23
HELIX    3   3 SER A  354  ALA A  371  1                                  18
HELIX    4   4 GLY A  398  ALA A  411  1                                  14
HELIX    5   5 SER A  420  ILE A  425  1                                   6
.
.
SHEET    1   A 5 ARG A 339  GLU A 340  0
SHEET    2   A 5 VAL A 349  PHE A 352 -1  O  THR A 351   N  ARG A 339
SHEET    3   A 5 ARG A 293  ILE A 303 -1  N  MET A 299   O  ALA A 350
SHEET    4   A 5 VAL A 376  ARG A 388 -1  O  HIS A 381   N  ILE A 298
.
.
.
LINK        MN    MN A 501                OG  SER A 306     1555   1555
2.61
.
.
.
```

```
SCALE3      0.000000  0.000000  0.014663        0.00000
.
.
.
ATOM      1  N   ALA A 291      -3.217  50.150   9.178  1.00 87.17
N
ATOM      2  CA  ALA A 291      -2.023  50.600   9.957  1.00 89.02
C
ATOM      3  C   ALA A 291      -1.678  52.060   9.658  1.00 87.59
C
ATOM      4  O   ALA A 291      -2.191  52.647   8.701  1.00 88.71
O
ATOM      5  CB  ALA A 291      -0.824  49.707   9.635  1.00 88.22
C
ATOM      6  N   GLU A 292      -0.808  52.634  10.486  1.00 82.15
N
ATOM      7  CA  GLU A 292      -0.370  54.017  10.331  1.00 78.31
C
ATOM      8  C   GLU A 292       1.086  54.119  10.734  1.00 71.89
C
ATOM      9  O   GLU A 292       1.572  53.284  11.492  1.00 71.66
O
ATOM     10  CB  GLU A 292      -1.171  54.948  11.238  1.00 82.93
C
ATOM     11  CG  GLU A 292      -2.658  54.982  10.979  1.00 90.28
C
ATOM     12  CD  GLU A 292      -3.380  55.853  11.987  1.00 95.55
C
ATOM     13  OE1 GLU A 292      -3.294  55.555  13.201  1.00 98.03
O
ATOM     14  OE2 GLU A 292      -4.029  56.837  11.569  1.00 98.67
O
ATOM     15  N   ARG A 293       1.790  55.125  10.218  1.00 64.48
N
ATOM     16  CA  ARG A 293       3.182  55.308  10.610  1.00 57.90
C
ATOM     17  C   ARG A 293       3.085  56.068  11.930  1.00 56.23
C
ATOM     18  O   ARG A 293       2.530  57.162  11.974  1.00 57.90
O
ATOM     19  CB  ARG A 293       3.946  56.147   9.583  1.00 51.01
C
ATOM     20  CG  ARG A 293       5.463  56.074   9.719  1.00 47.57
C
ATOM     21  CD  ARG A 293       6.001  54.783   9.128  1.00 51.26
C
ATOM     22  NE  ARG A 293       7.448  54.626   9.272  1.00 45.82
N
ATOM     23  CZ  ARG A 293       8.338  55.532   8.895  1.00 39.47
C
ATOM     24  NH1 ARG A 293       7.925  56.662   8.352  1.00 42.49
N
```

```
ATOM     25  NH2 ARG A 293       9.635  55.309   9.065  1.00 41.43
N
ATOM     26  N   MET A 294       3.594  55.481  13.009  1.00 51.83
N
ATOM     27  CA  MET A 294       3.534  56.138  14.305  1.00 42.82
C
ATOM     28  C   MET A 294       4.890  56.200  14.980  1.00 41.75
C
ATOM     29  O   MET A 294       5.727  55.312  14.806  1.00 41.60
O
ATOM     30  CB  MET A 294       2.564  55.417  15.220  1.00 45.73
C
ATOM     31  CG  MET A 294       1.149  55.314  14.687  1.00 50.34
C
ATOM     32  SD  MET A 294      -0.028  54.803  15.972  1.00 57.91
S
ATOM     33  CE  MET A 294       1.062  53.800  17.041  1.00 49.13
C
ATOM     34  N   LEU A 295       5.110  57.269  15.742  1.00 39.09
N
ATOM     35  CA  LEU A 295       6.362  57.442  16.474  1.00 31.61
C
ATOM     36  C   LEU A 295       6.225  56.575  17.720  1.00 24.45
C
ATOM     37  O   LEU A 295       5.174  56.578  18.365  1.00 17.45
O
ATOM     38  CB  LEU A 295       6.547  58.899  16.892  1.00 27.73
C
ATOM     39  CG  LEU A 295       7.968  59.459  16.859  1.00 31.15
C
ATOM     40  CD1 LEU A 295       8.022  60.654  17.795  1.00 26.63
C
ATOM     41  CD2 LEU A 295       8.990  58.417  17.276  1.00 17.64
C
ATOM     42  N   ALA A 296       7.268  55.823  18.054  1.00 20.35
N
ATOM     43  CA  ALA A 296       7.194  54.961  19.230  1.00 23.28
C
ATOM     44  C   ALA A 296       8.539  54.645  19.801  1.00 18.33
C
ATOM     45  O   ALA A 296       9.565  54.759  19.140  1.00 21.17
O
ATOM     46  CB  ALA A 296       6.458  53.649  18.896  1.00 31.20
C
ATOM     47  N   THR A 297       8.531  54.282  21.070  1.00 24.69
N
ATOM     48  CA  THR A 297       9.766  53.894  21.742  1.00 28.20
C
ATOM     49  C   THR A 297       9.592  52.402  21.934  1.00 26.43
C
ATOM     50  O   THR A 297       8.590  51.932  22.499  1.00 22.03
O
```

```
ATOM     51  CB  THR A 297       9.941  54.612  23.088  1.00 28.57
C
ATOM     52  OG1 THR A 297       9.856  56.027  22.874  1.00 29.82
O
ATOM     53  CG2 THR A 297      11.309  54.286  23.676  1.00 15.05
C
.
.
.
HETATM 5166  O   HOH D  85      23.083  60.473  73.054  1.00 59.91
O
HETATM 5167  O   HOH D  88      22.733  31.412  53.477  1.00 46.50
O
HETATM 5168  O   HOH D  94      25.211  35.976  69.262  1.00 18.16
O
HETATM 5169  O   HOH D 103      23.392  28.639  57.541  1.00 38.29
O
HETATM 5170  O   HOH D 107      30.841  43.237  73.843  1.00 50.65
O
HETATM 5171  O   HOH D 110      26.181  62.701  70.920  1.00 42.52
O
HETATM 5172  O   HOH D 111      25.347  30.761  61.451  1.00 41.68
O
HETATM 5173  O   HOH D 114      32.500  73.206  60.330  1.00 47.72
O
HETATM 5174  O   HOH D 117      31.682  38.936  47.332  1.00 19.73
O
HETATM 5175  O   HOH D 121      31.730  54.329  63.102  1.00 31.75
O
HETATM 5176  O   HOH D 122      27.932  26.904  69.142  1.00 44.00
O
HETATM 5177  O   HOH D 125      22.473  44.995  71.200  1.00 58.27
O
HETATM 5178  O   HOH D 138      25.895  28.112  64.181  1.00 18.84
O
MASTER      538    0    4   20   48    0   11    6 5191    4   73   60
END
```

Lampiran B

Source Code Training IDELM

```
train_data=TrainingData_File;
test_data=TestingData_File;
T=train_data(:,size(train_data,2));
P=train_data(:,1:size(train_data,2)-1);
TV.T=test_data(:,size(test_data,2));
TV.P=test_data(:,1:size(test_data,2)-1);
clear train_data test_data;

nTrainingData=size(P,1);
nTestingData=size(TV.P,1);
nInputNeurons=size(P,2);

%%%%%%%%%%%% Preprocessing T in the case of CLASSIFICATION

start_time_train=cputime;
%%%%%%%%%%%% step 1 Initialization Phase
P0=P(1:N0,:);
T0=T(1:N0,:);

IW = rand(nHiddenNeurons,nInputNeurons)*2-1;
Bias = rand(1,nHiddenNeurons)*2-1;
H0 = SigActFun(P0,IW,Bias);
M = pinv(H0' * H0);
beta = pinv(H0) * T0;
Y0=H0*beta;
e0=T0-Y0;
ms0=mse(e0);
%%%%%%%%%%%%%% step 2 Sequential Learning Phase
for n = N0 : Block : nTrainingData
    iter=n;
    if (n+Block-1) > nTrainingData
        Pn = P(n:nTrainingData,:) ;
        Tn = T(n:nTrainingData,:);
        Block = size(Pn,1);             %%%% correct the block size
        clear V;                        %%%% correct the first dimention
of V
    else
        Pn = P(n:(n+Block-1),:);
        Tn = T(n:(n+Block-1),:);
    end
%     IW
%     Bias
    H = SigActFun(Pn,IW,Bias);
    cbeta= pinv(H) * Tn;
    Y1=H*cbeta;

    e=Tn-Y1;
    ms=mse(e);
    delta=ms-ms0;
ms0=ms;
if delta<0
disp('hapus......');
ms=ms0;
else
```

```
    M = M - M * H' * (eye(Block) + H * M * H')^(-1) * H * M ;
    beta = beta + M * H' * (Tn - H * beta);
end
end
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train;
% clear Pn Tn H M;

HTrain = SigActFun(P, IW, Bias);
Y=HTrain * beta;
clear HTrain;
```

Lampiran C

Source Code Testing IDELM dan Perhitungan Performa

```
start_time_test=cputime;

HTest = SigActFun(TV.P, IW, Bias);

TY=HTest * beta;
clear HTest;
end_time_test=cputime;
TestingTime=end_time_test-start_time_test;
[c k]=size(TY);


for i=1:c
 if TY(i)>0.4;
     TY(i)=1;
 else
     TY(i)=0;
 end
 end
 Yelm=TY;
Yout=Yelm';
cm=confusionmat(TV.T,Yelm);
if size(cm)==[1 1]
    if sum(Yelm)>0
        b=zeros(2,2);
        b(2,2)=cm;
    else sum(Yelm)<=0
        b=zeros(2,2);
        b(1,1)=cm;
    end
else
        b=cm;
end
% figure,plotconfusion(t,handles.G)
TRUEP=b(2,2)
FALSEN=b(2,1)
TRUEN=b(1,1)
FALSEP=b(1,2)
TOTAL=TRUEP+TRUEN+FALSEP+FALSEN;
accuracy1=(TRUEP+TRUEN)/TOTAL;
precission1=TRUEP/(TRUEP+FALSEP);
recall1=TRUEP/(TRUEP+FALSEN);
specify1=1-(FALSEP/(FALSEP+TRUEN));
gmean1=sqrt(recall1*specify1);
```

Lampiran D

Source Code Fungsi untuk Menghitung Matriks Hidden Layer

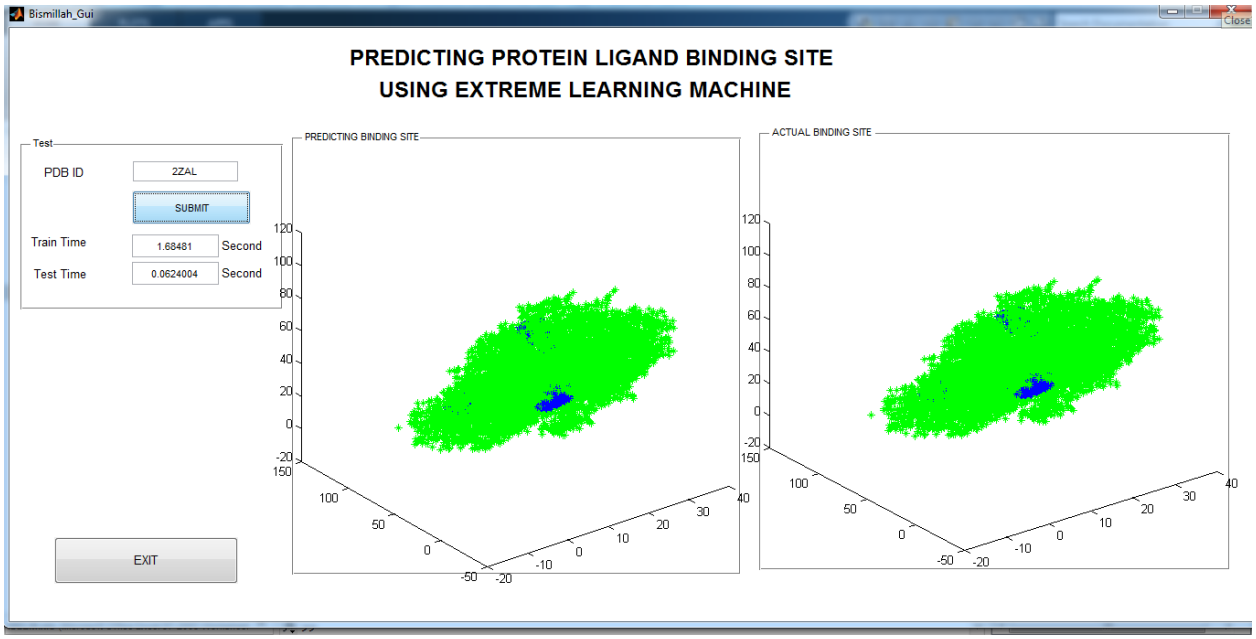
```
function H = SigActFun(P,IW,Bias);

%%%%%%%% Feedforward neural network using sigmoidal activation function
V=P*IW';
ind=ones(1,size(P,1));
BiasMatrix=Bias(ind,:);
V=V+BiasMatrix;
H = 1./(1+exp(-V));
```

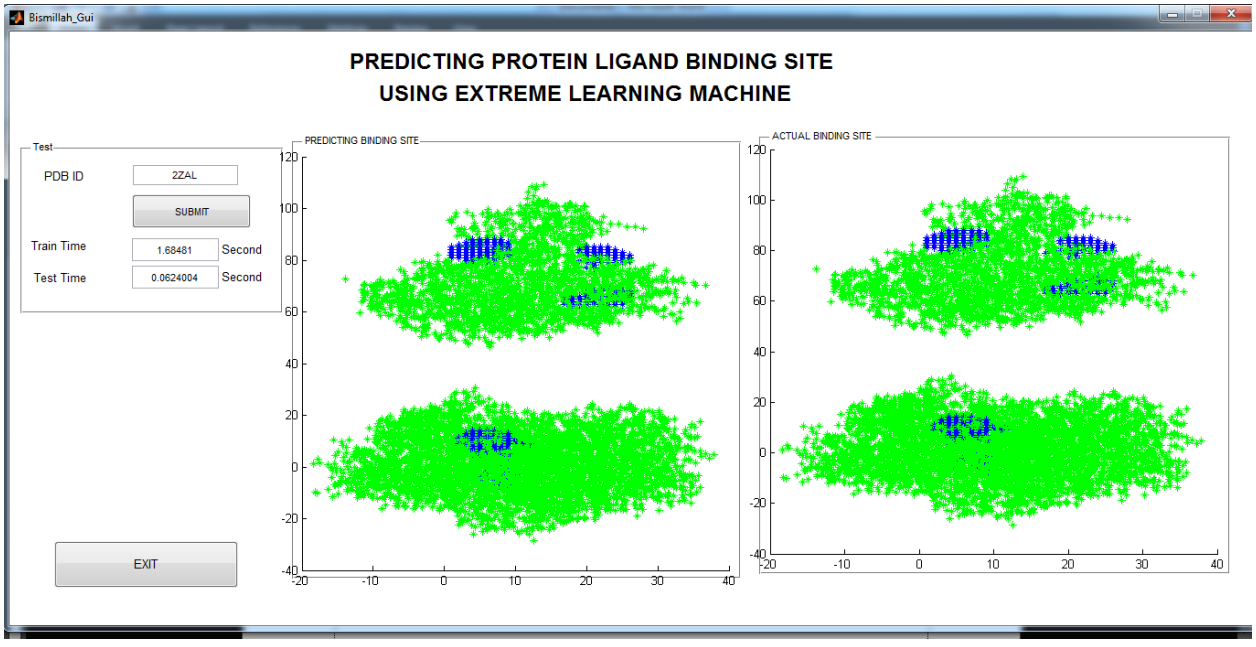
Lampiran E

Tampilan Hasil Plot Beberapa Protein

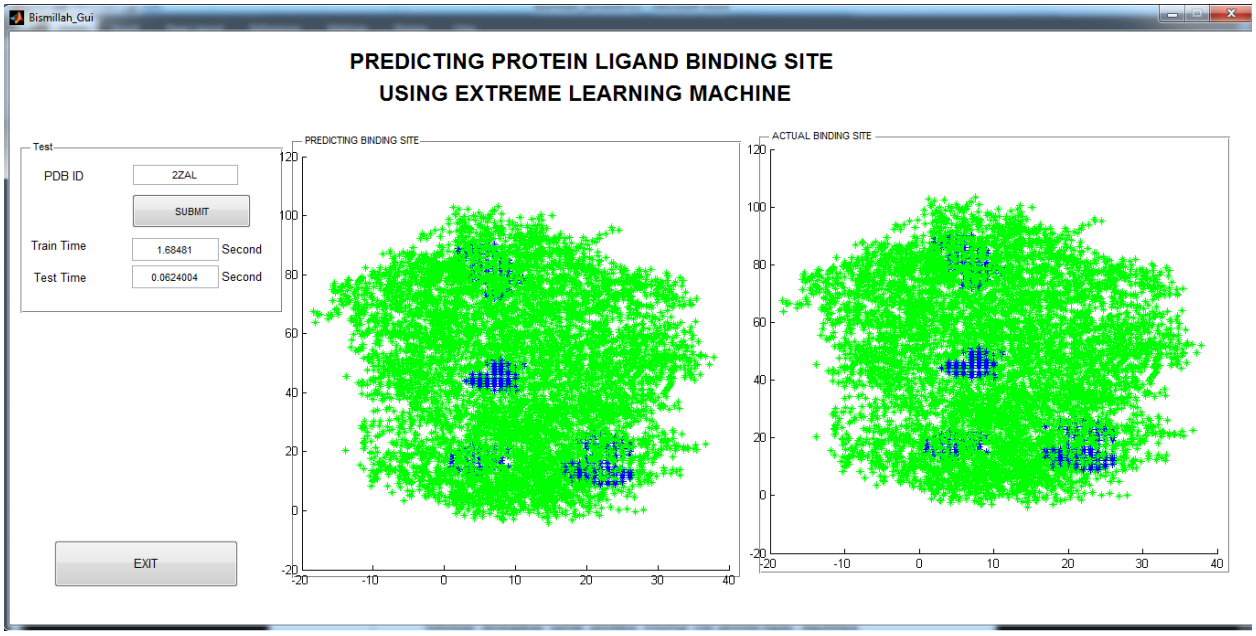
2ZAL plot 3D



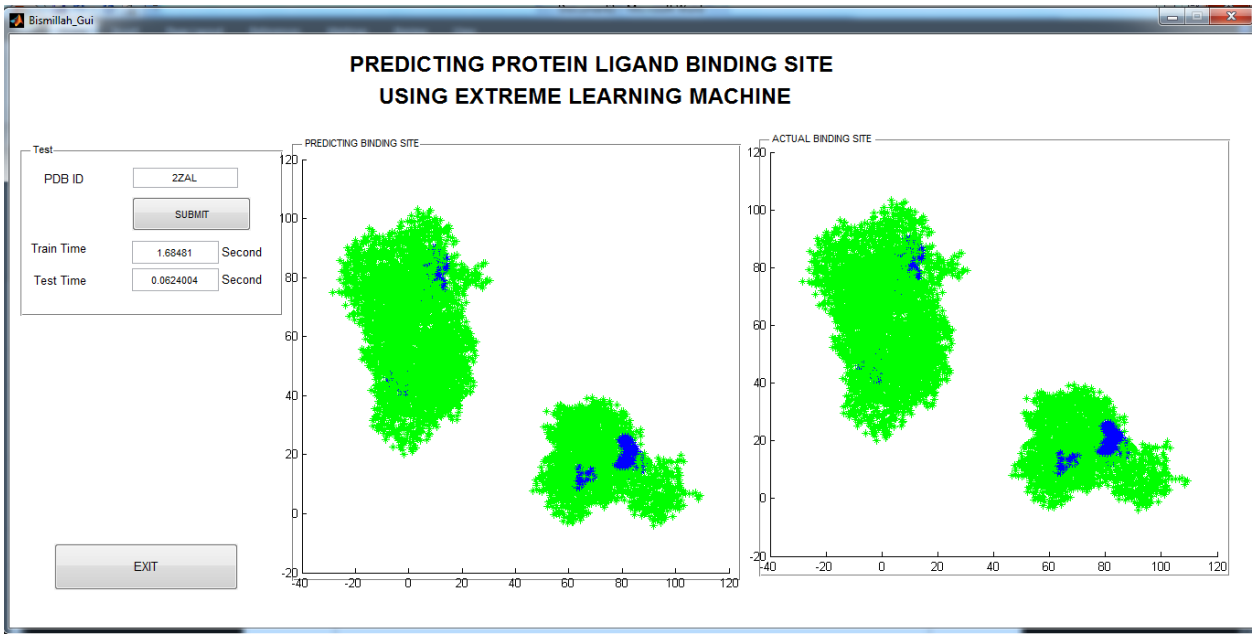
2ZAL lilihat dari sumbu Z



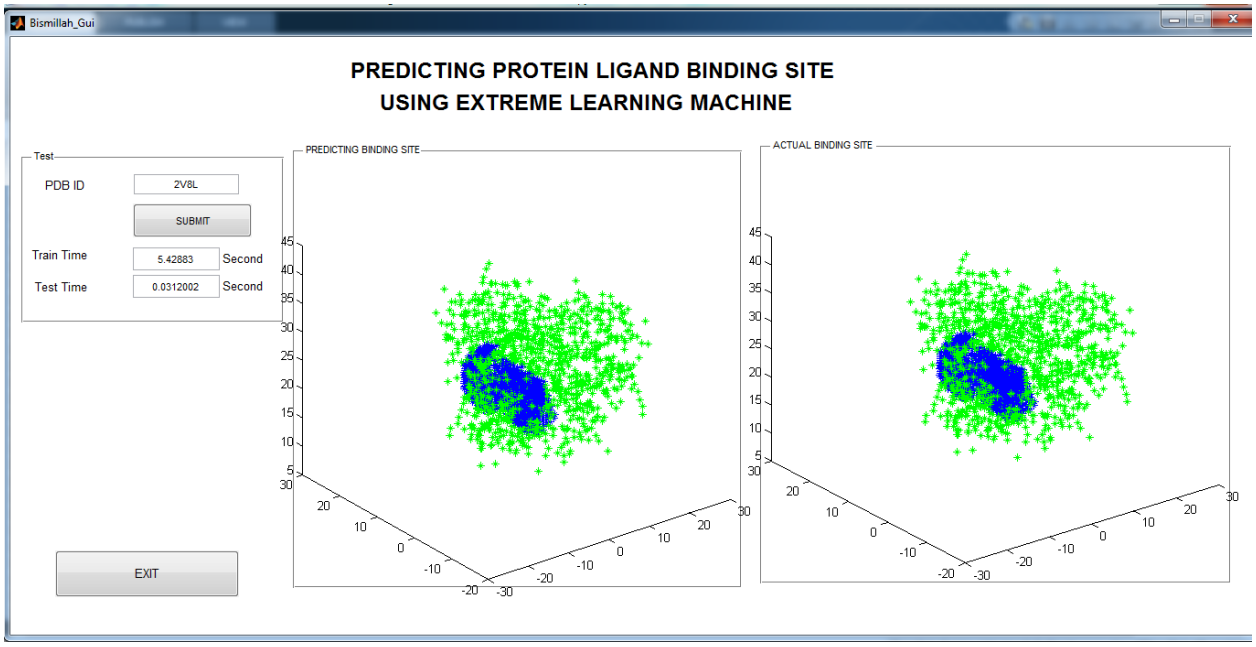
2ZAL lilihat dari sumbu Y



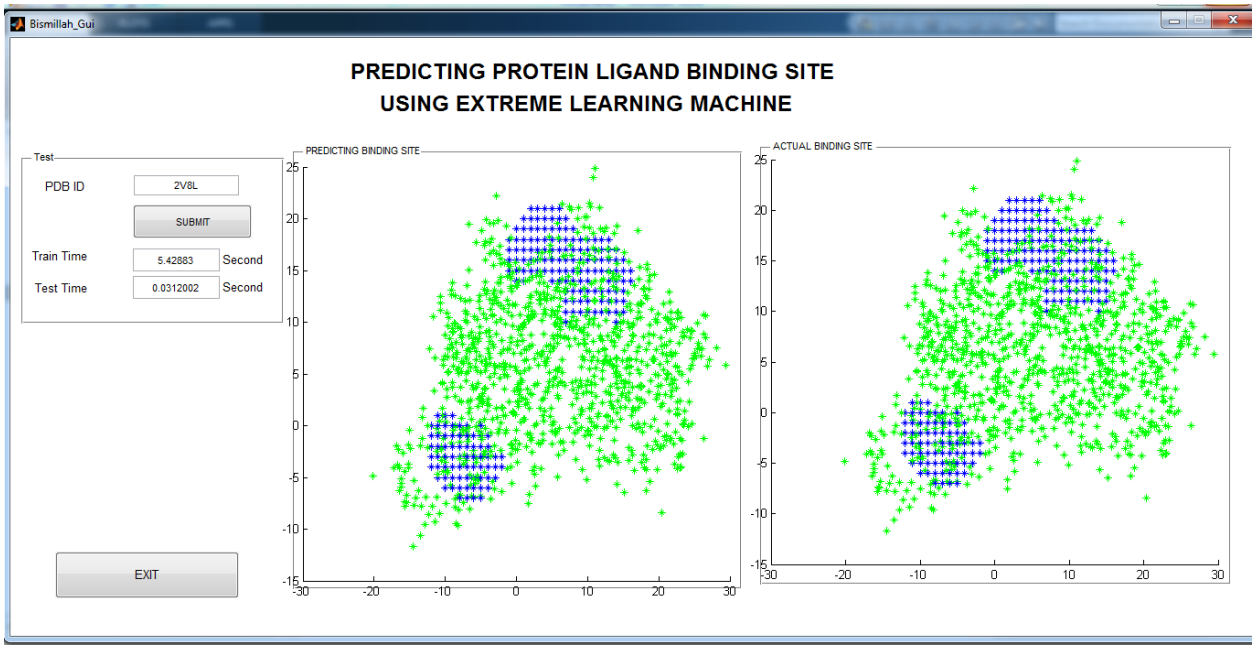
2ZAL lilihat dari sumbu X



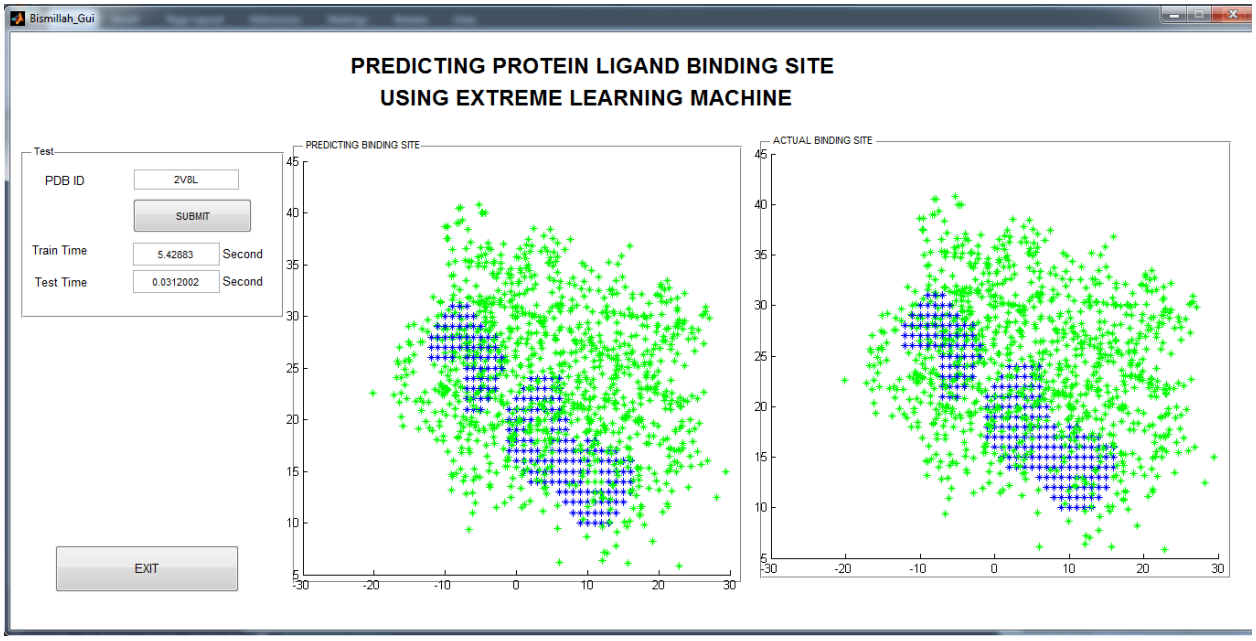
Plot 3D 2V8L



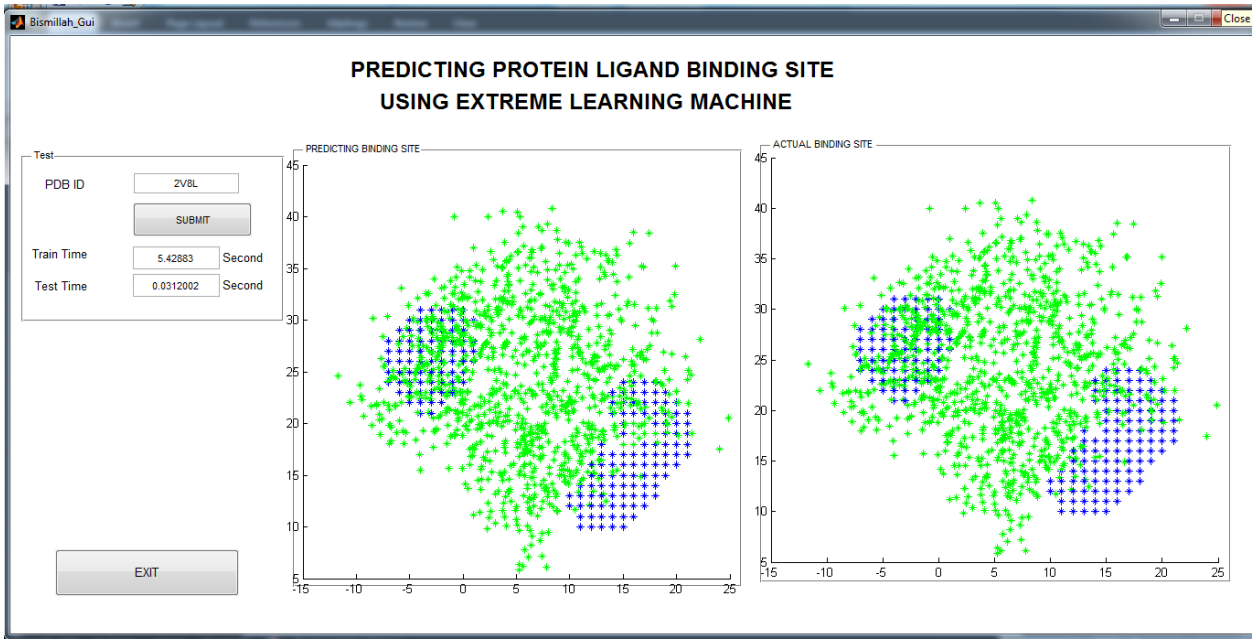
2V8L dilihat dari sumbu Z



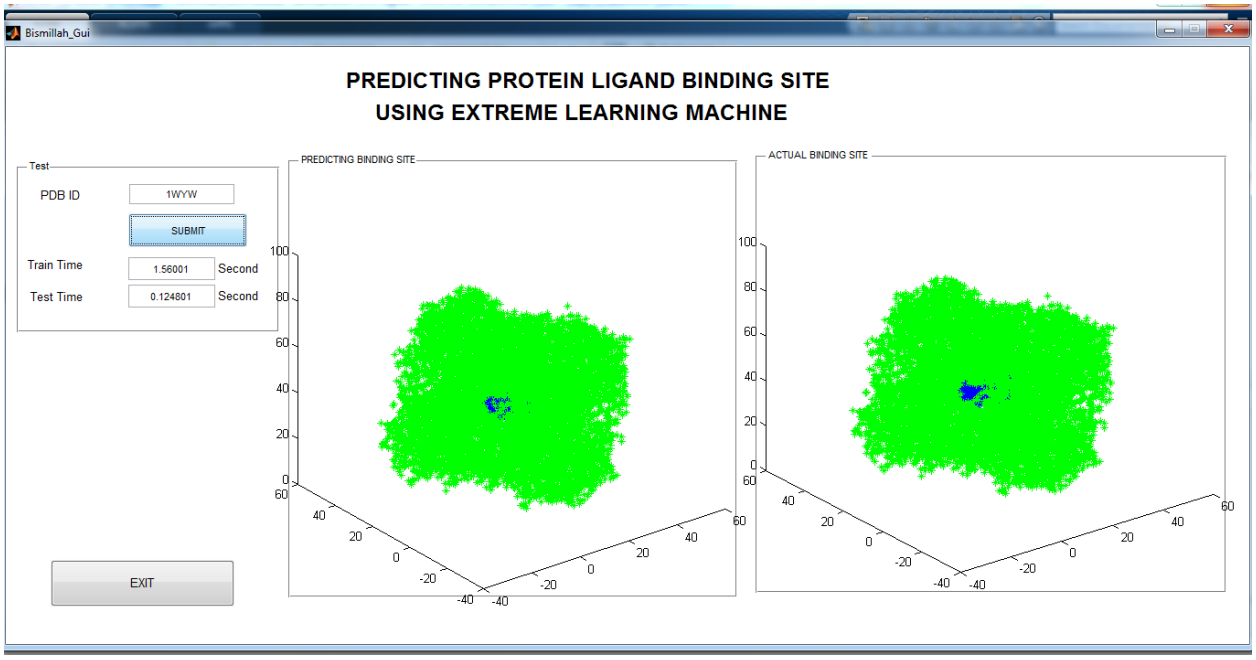
2V8L dilihat dari sumbu Y



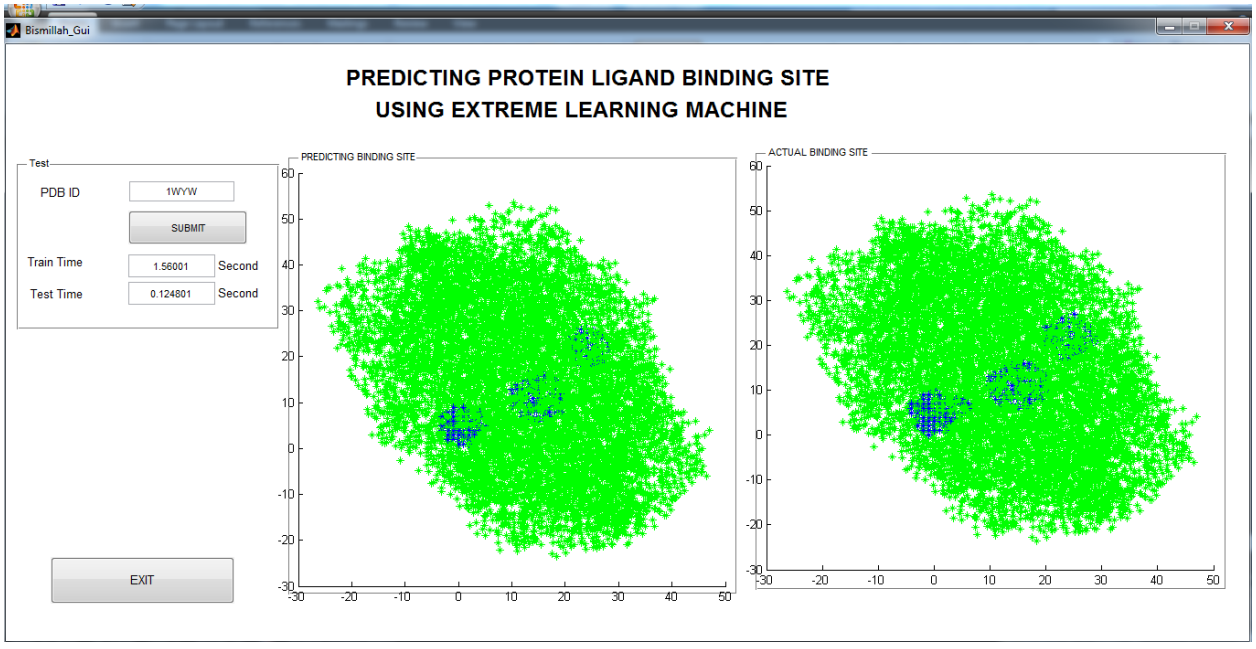
2V8L dilihat dari sumbu X



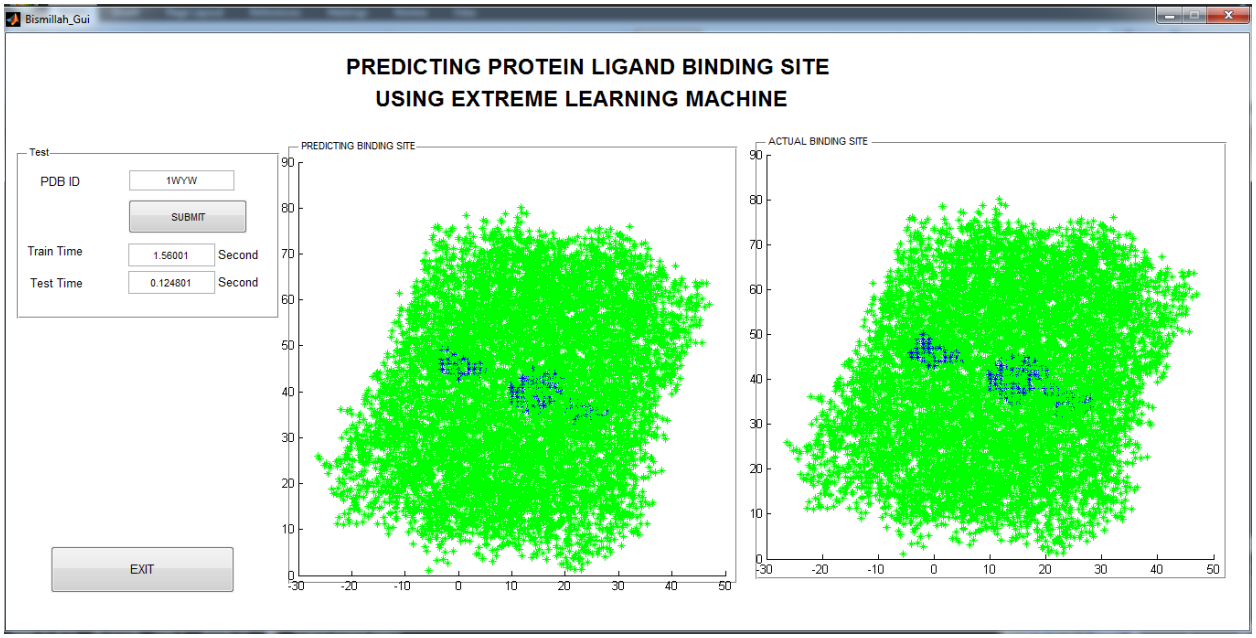
1WYW



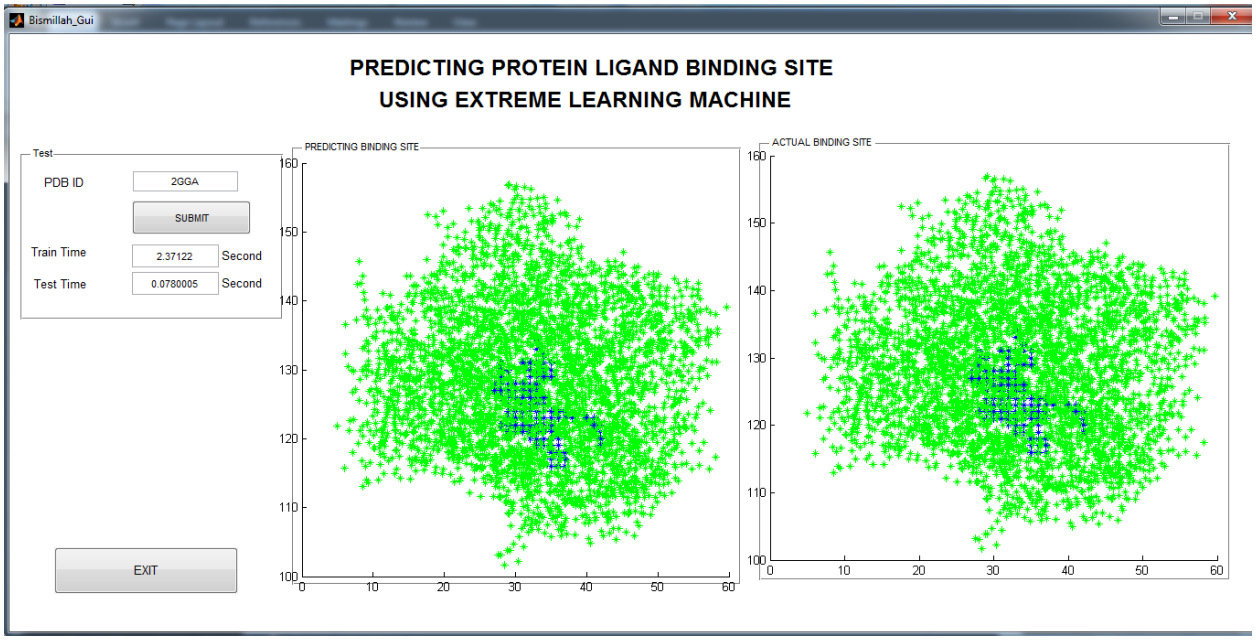
1WYW dilihat dari sumbu Z



1WYW dilihat dari sumbu Y



1WYW dilihat dari sumbu X



Lampiran F

Hasil Akurasi IDELM Pada Setiap Proses Training

Tipe	ID	TRAIN 1	TRAIN 2	TRAIN 3	TRAIN 4	TRAIN 5	TRAIN 6	Rata - rata	SD
Hydrolase	4TPI	0.971729	0.966469	0.982906	0.965812	0.980934	0.991782	0.976605	0.010313
	2ZAL	0.988649	0.994135	0.97465	0.984466	0.984466	0.945631	0.978666	0.017399
	2V8L	0.976214	0.959223	0.966505	0.987864	0.984466	0.96165	0.972654	0.012023
	1WYW	0.989705	0.802101	0.932404	0.880616	0.914725	0.989913	0.918244	0.071219
	1RN8	0.978504	0.962382	0.962382	0.98567	0.987909	0.982535	0.976564	0.01143
	1C1P	0.988743	0.944757	0.993955	0.970398	0.992287	0.996248	0.981065	0.020068
	1YBU	0.992046	0.767812	0.993738	0.984431	0.906752	0.947368	0.932025	0.087175
Oxidoreductase	3D4P	0.991779	0.992709	-	-	-	-	0.992244	0.000657
	1A4U	0.983916	0.955822	-	-	-	-	0.969869	0.019865
	2WLA	0.946399	0.940262	-	-	-	-	0.943331	0.00434
Transferase	2GGA	0.999035	0.994211	0.99807	0.995658	-	-	0.996744	0.002206
	1SQF	0.984651	0.98488	0.966323	0.987858	-	-	0.980928	0.009846
	1O26	0.889668	0.885246	0.754853	0.993421	-	-	0.880797	0.097715
	1G6C	0.993561	0.989565	0.986456	0.990897	-	-	0.99012	0.002954
	1BJ4	0.737693	0.738644	0.989061	0.989774	-	-	0.863793	0.145059
Ligase	1U7Z	0.971517	-	-	-	-	-	0.971517	
	1ADE	0.994819	-	-	-	-	-	0.994819	

Lampiran F (Lanjutan)

Hasil *Recall* IDELM Pada Setiap Proses Training

Tipe	ID	TRAIN 1	TRAIN 2	TRAIN 3	TRAIN 4	TRAIN 5	TRAIN 6	Rata - rata	SD
Hydrolase	4TPI	0.9601	0.9755	0.9330	0.9678	0.9729	0.9678	0.9628	0.0156
	2ZAL	0.9128	0.9549	0.8052	0.9738	0.9699	0.8913	0.9180	0.0642
	2V8L	0.9524	0.9854	0.9330	0.9777	0.9874	0.9233	0.9599	0.0277
	1WYW	0.8803	0.9178	0.7630	0.8162	0.9625	0.8827	0.8704	0.0713
	1RN8	0.9477	0.9858	0.9858	0.9684	0.9739	0.9575	0.9699	0.0153
	1C1P	0.9234	0.9702	0.9589	0.7986	0.9504	0.9787	0.9300	0.0672
	1YBU	0.9345	0.9958	0.9485	0.8719	0.9721	0.7925	0.9192	0.0749
Oxidoreductase	3D4P	0.9504	0.9634	-	-	-	-	0.9569	0.0092
	1A4U	0.9362	0.7910	-	-	-	-	0.8636	0.1027
	2WLA	0.8487	0.8357	-	-	-	-	0.8422	0.0092
Transferase	2GGA	0.9873	0.9968	0.9968	0.9430	-	-	0.9810	0.0257
	1SQF	0.9347	0.9347	0.9058	0.9486	-	-	0.9309	0.0180
	1O26	0.9384	0.9437	0.9369	0.9542	-	-	0.9433	0.0078
	1G6C	0.9599	0.9351	0.9157	0.9434	-	-	0.9385	0.0184
	1BJ4	0.9581	0.9287	0.9036	0.9099	-	-	0.9251	0.0245
Ligase	1U7Z	0.7606	-	-	-	-	-	0.7606	
	0.938953	0.9416	-	-	-	-	-	0.9416	

Lampiran F (Lanjutan)

Hasil *Specificity* IDELM Pada Setiap Proses Training

Tipe	ID	TRAIN 1	TRAIN 2	TRAIN 3	TRAIN 4	TRAIN 5	TRAIN 6	Rata - rata	SD
Hydrolase	4TPI	0.9757	0.9634	1.0000	0.9651	0.9837	1.0000	0.9813	0.0162
	2ZAL	1.0000	1.0000	1.0000	0.9951	0.9990	1.0000	0.9990	0.0019
	2V8L	1.0000	0.9330	1.0000	0.9981	0.9816	1.0000	0.9854	0.0267
	1WYW	1.0000	0.7912	0.9483	0.8867	0.9102	1.0000	0.9227	0.0792
	1RN8	1.0000	0.9460	0.9460	0.9977	0.9977	1.0000	0.9812	0.0273
	1C1P	1.0000	0.9404	1.0000	1.0000	0.9995	0.9993	0.9899	0.0242
	1YBU	1.0000	0.7363	1.0000	1.0000	0.8977	0.9688	0.9338	0.1046
Oxidoreductase	3D4P	0.9996	0.9982	-	-	-	-	0.9989	0.0010
	1A4U	0.9929	0.9868	-	-	-	-	0.9898	0.0043
	2WLA	0.9981	0.9956			-	-	0.9969	0.0018
Transferase	2GGA	1.0000	0.9940	0.9982	1.0000	-	-	0.9980	0.0028
	1SQF	0.9983	0.9985	0.9828	0.9985	-	-	0.9945	0.0078
	1O26	0.8815	0.8754	0.7243	1.0000	-	-	0.8703	0.1130
	1G6C	1.0000	1.0000	1.0000	1.0000	-	-	1.0000	0.0000
	1BJ4	0.7095	0.7143	1.0000	1.0000	-	-	0.8560	0.1663
Ligase	1U7Z	1.0000	-	-	-	-	-	1.0000	
	1ADE	1.0000	-	-	-	-	-	1.0000	

Lampiran F (Lanjutan)

Hasil *G-mean* IDELM Pada Setiap Proses Training

Tipe	ID	TRAIN 1	TRAIN 2	TRAIN 3	TRAIN 4	TRAIN 5	TRAIN 6	Rata -rata	SD
Hydrolase	4TPI	0.9679	0.9694	0.9659	0.9665	0.9783	0.9838	0.9720	0.0073
	2ZAL	0.9554	0.9772	0.8973	0.9844	0.9844	0.9441	0.9571	0.0336
	2V8L	0.9759	0.9589	0.9659	0.9878	0.9845	0.9609	0.9723	0.0123
	1WYW	0.9382	0.8521	0.8506	0.8507	0.9360	0.9395	0.8945	0.0475
	1RN8	0.9735	0.9657	0.9657	0.9830	0.9857	0.9785	0.9754	0.0085
	1C1P	0.9609	0.9552	0.9792	0.8936	0.9746	0.9889	0.9588	0.0342
	1YBU	0.9667	0.8563	0.9739	0.9337	0.9342	0.8762	0.9235	0.0477
Oxidoreductase	3D4P	0.9747	0.9806					0.9777	0.0042
	1A4U	0.9641	0.8835	-	-	-	-	0.9238	0.0570
	2WLA	0.9204	0.9122	-	-	-	-	0.9163	0.0058
Transferase	2GGA	0.9937	0.9954	0.9975	0.9711	-	-	0.9894	0.0123
	1SQF	0.9659	0.9661	0.9435	0.9733	-	-	0.9622	0.0129
	1O26	0.9095	0.9089	0.8238	0.9768	-	-	0.9048	0.0627
	1G6C	0.9798	0.9670	0.9569	0.9713	-	-	0.9687	0.0095
	1BJ4	0.8245	0.8145	0.9506	0.9539	-	-	0.8858	0.0768
Ligase	1U7Z	0.8721	-	-	-	-	-	0.8721	
	0.968996	0.9704	-	-	-	-	-	0.9704	

Lampiran F (Lanjutan)

Hasil *CPU Time* IDELM Pada Setiap Proses Training

Tipe	ID	TRAIN 1 (s)	TRAIN 2 (s)	TRAIN 3 (s)	TRAIN 4 (s)	TRAIN 5 (s)	TRAIN 6 (s)	Rata –rata (s)	SD
Hydrolase	4TPI	1.0452	1.0764	5.6784	1.2480	2.5584	3.2448	2.4752	1.8093
	2ZAL	1.6848	1.1700	5.6004	1.0764	2.5740	2.9328	2.5064	1.6891
	2V8L	1.5912	2.8392	5.1792	1.1544	2.4804	3.4008	2.7742	1.4346
	1WYW	1.5600	1.8252	1.0140	1.2948	2.8236	3.4008	1.9864	0.9309
	1RN8	1.4508	2.9640	2.9640	2.7300	2.6832	3.3540	2.6910	0.6524
	1C1P	1.7316	2.9484	1.0764	5.3196	1.2324	3.1824	2.5818	1.6000
	1YBU	1.5444	2.7768	1.2636	5.2104	1.2324	2.5584	2.4310	1.5130
Oxidoreductase	3D4P	2.6676	1.3104	-	-	-	-	1.9890	0.9597
	1A4U	3.5880	1.2012	-	-	-	-	2.3946	1.6877
	2WLA	2.6520	2.5740	-	-	-	-	2.6130	0.0552
Transferase	2GGA	2.3712	2.4960	2.6832	2.2152	-	-	2.4414	0.1979
	1SQF	2.4648	5.0544	2.4960	2.1840	-	-	3.0498	1.3438
	1O26	2.3712	2.2776	2.5584	2.2464	-	-	2.3634	0.1404
	1G6C	2.2464	2.3868	5.1012	2.3244	-	-	3.0147	1.3922
	1BJ4	2.4804	2.5740	5.4600	5.1168	-	-	3.9078	1.6008
Ligase	1U7Z	4.8516	-	-	-	-	-	4.8516	-
	1ADE	3.2916	-	-	-	-	-	3.2916	-

Lampiran G

Target dan Hasil Klasifikasi Protein 2ZAL

Output dari Integrasi seleksi data dan *Extreme Learning Machine* untuk prediksi *binding site* dari protein 2ZAL
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0, 0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,0,0,0,0,1, 0,0,0,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,0, 0,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,0,1,1,1,0,0, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1, 1,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,0,0,0,1,0,0,0,1,1,1,0,0,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,0,1,0,0,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,1,1,1,1,0,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0, 0,1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1, 0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,0,0, 1,1,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1, 1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 0,0,0,1,1,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0
Target dari protein 2ZAL
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0
```

Lampiran

Paper Publikasi Ilmiah

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2015) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)

Integrating Data Selection and Extreme Learning Machine for Imbalanced Data

Umi Mahdiyah[a,*], M. Isa Irawan[a], Elly Matul Imah[b]

[a]*Sepuluh Nopember Institute of Technology, Keputih, Surabaya, 60111, Indonesia*
[b]*The State University of Surabaya, Ketintang, Surabaya, 50231, Indonesia*

Abstract

Extreme Learning Machine (ELM) is one of the artificial neural network method that introduced by Huang, this method has very fast learning capability. ELM is designed for balance data. Common problems in real-life is imbalanced data problem. So, for imbalanced data problem needs special treatment, because characteristics of the imbalanced data can decrease the accuracy of the data classification. The proposed method in this study is modified ELM to overcome the problems of imbalanced data by integrating the data selection process, which is called by Integrating the data selection and extreme learning machine (IDELM. Performances of learning method are evaluated using 13 imbalanced data from UCI Machine Learning Repository and Benchmark Data Sets for Highly Imbalanced Binary Classification (BDS). The validation includes comparison with some learning algorithms and the result showcases that average perform of our proposed learning method is compete and even outperform of some algorithm in some cases.

Peer-review under responsibility of organizing committee of the International Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

Keywords: Data Selection; Extreme Learning Machine; Imbalanced Data

1. Introduction

A successful understanding of how to make computers learn would open up many new uses of computers and new levels of competence and customization, so widely used in people lives[3,4,15]. The detailed understanding of information- processing algorithms for machine learning might lead to a better understanding of human learning abilities (and disabilities) as well[1]. Many type of machine learning that we know, one of them is extreme learning machine (ELM). This method have very fast learning than Back propagation and Support Vector Machine[2].

Extreme learning machine is one of a new learning algorithm in neural networks, which has the single-hidden layer feed-forward network (SLFN). ELM has a very fast learning capability[4]. First ELM was introduced by Huang as single-hidden layer feed-forward network. ELM was made to overcome the weaknesses learning speed of the feed-forward neural networks problem. Traditionally, gradient-based learning algorithm is used for feed-forward neural

* Corresponding author.
E-mail address: umimahdiyah@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

network training, all the parameters (input weight and hidden bias) are determined by iterative network, to solve that problem, Extreme Learning Machine uses minimum norm least-squares (LS) solution of SLFNs.

Unlike the traditional function approximation theories which require to adjusted input weights and hidden layer biases, input weights and hidden layer biases can be randomly assigned if only the activation function is infinitely differentiable[5]. So, ELM can be faster than prior the neural network algorithm previously. ELM has been applied and developed in various fields[6,7,8,9,10].Up until now this algorithm is developed, the main reason many research use this algorithm because this algorithm is simple and faster than several algorithm in neural network.

Imbalanced dataset is a problem which is one of the important issues in classification problems and become a new challenge in machine learning recently[11].Raw data with imbalanced class distribution can be found almost every real world problem, include medical problem, security, internet, finance and etc. When classifying data with imbalanced class distribution, the regular learning algorithm has a natural tendency to favor the majority class by assuming balanced class distribution or equal misclassification cost[12]. To solving imbalanced data problem there are several ways, that undersampling, oversampling and modification algorithms[13]. In previous studies, there are several ways to solve the problem of imbalanced data with the modification of the algorithm, some of which is the hybrid algorithm[14] and integrating several step in learning process[15,16,17,18].

In this paper will be proposed to solving imbalanced data problem using data selection during the training process. If the data selection and classification process is done separately, there will be inconsistencies between these two steps. Thus, to overcome these problems, in this paper also proposed the integration of data selection and classification steps with Extreme Learning Machine.

The rest of the paper is organized as follows. In Section 2, we present detail ELM. In Section 3, we present detail about integrating data selection and extreme learning machine (IDELM) for imbalanced data. Section 4. we describe dataset and experimental settings. Following that, Section 5 provides experimental results and discussions. Finally, we draw conclusions in Section 6.

2. Extreme Learning Machine

Extreme learning machine is one of a new learning algorithm in neural networks, which has the Single-hidden Layer feed-forward Network (SLFN)[4,5]. ELM has a simple algorithm, very fast learning capability and training small error. First ELM was introduced by Huang as single-hidden layer feed-forward network (SLFNs). ELM made to overcome the weaknesses of the feed-forward neural networks problem that learning speed. Traditionally, feed-forward neural network using gradient-based learning algorithm for training, as well as all the parameters (input weight and hidden bias) are determined by iterative network, to solve that problem, extreme learning machine using minimum norm least-squares (LS) solution of SLFNs.

A standard single layer feedforward neural network with n hidden neurons and activation function $g(x)$ can be mathematically modelled as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i x_j + b_i) = o_j, j \in [1, N] \tag{1}$$

Which $w_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is input weight.b_i is bias hidden layer. $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{in}]^T$ output weight, and $w_i x_i$ is inner product from w_i and x_i, then o_j is output from this algorithm.

Standard SLFN with N hidden nodes and activation function $g(x)$ can approximate these N samples with zero error means that if

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i x_j + b_i) = t_j \tag{2}$$

That equation can be write as

$$H\beta_i = T \tag{3}$$

Note:

H is output matrix hidden layer.

$$H = \begin{pmatrix} g(w_i x_i + b_i) & \cdots & g(w_{\tilde{N}} x_i + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_i x_N + b_i) & \cdots & g(w_{\tilde{N}} x_N + b_{\tilde{N}}) \end{pmatrix} \tag{4}$$

$$\beta = [\beta_1^T, \cdots, \beta_{\tilde{N}}^T]^T \tag{5}$$

$$T = [t_1^T, \cdots, t_N^T]^T \tag{6}$$

ELM algorithm is derived from the minimum norm least squares solution SLFNs. Although, ELM is "generalized" of SLFN but hidden layer (feature mapping) of the ELM does not need to be tuned. Main concepts of ELM as presented by Huang2 as follow there: Given training set

$$\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i \in [1, N]\} \tag{7}$$

activation function $g(x)$, and the number of hidden nodes $\tilde{N}$

Step 1: insert random weights and biases w_i and$b_i, i \in [1, N]$

Step 2: Calculate the hidden layer output matrix H

Step 3: Calculate the output weights β

Which $T = [t_1, t_2, ..., t_N]^T$

$H^\dagger$is MoorePenrose Generalized Inverse

Then activation function in the ELM must be infinitely differential (i.e sigmoid function, RBF, sine, cosine, exponential, etc.). Many hidden node depend on the number of training samples is N $\tilde{N} < N$. Several methods can be used to compute the Moore-Penrose Generalized Inverse of H, which are orthogonal projection, orthogonalization method, iterative method, and singular value decomposition (SVD).

To determine the number of hidden nodes, in this paper uses the following algorithm [19]:

Given a training set $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i \in [1, N]\}$, activation fuction $g(x)$,

maximum node number $\tilde{N}_{max}$, and expected learning accuracy ϵ:

Initialization: Let $\tilde{N} = 0$ and residual error E=t,where $t = [t_1, t_2, ..., t_N]^T$

Learning step: While $\tilde{N} < \tilde{N}_{max}$ and $\|E\| > \epsilon$

a) increase by one the number of hidden node $\tilde{N} : \tilde{N} = \tilde{N} + 1$;
b) assign random nput weight $a_{\tilde{N}}$ and bias $b_{\tilde{N}}$ (or random centre $a_{\tilde{N}}$ and impact factor $b_{\tilde{N}}$)for new hidden node $\tilde{N}$
c) calculate the output weight $\beta_{\tilde{N}}$ for the new hidden node

$$\beta_{\tilde{N}} = \frac{E \cdot H_{\tilde{N}}^T}{H_{\tilde{N}} \cdot H_{\tilde{N}}^T} \tag{8}$$

d) calculate the residual error after adding the new hidden node $\tilde{N}$:

$$E = E - \beta_{\tilde{N}} . H_{\tilde{N}} \tag{9}$$

endwhile

3. Integrating Data Selection and Extreme Learning Machine

Integrating data selection and classification is an improvement ELM, which is designed to be able to perform data selection and classification as well. To integrate that step, in this paper used the MSE, because MSE is often used for stopping condition in Extreme Learning Machine.

The first process of this proposed method is to divide the training data into several parts. The first group of data is used to initialize the number of hidden nodes and initial MSE. The next group, after entering the learning process also calculated its MSE. Then, when MSE in next group (data-t + 1) is higher than the previous group (data-t), then the data entry to selected data. Conversely, if the next data is lower than the previous data (t), the data is not used. Because, the MSE in data-t + 1 is greater than the data- t then there is a unique data on the data-t + 1. The block diagram of the integration of learning methods can be seen in Figure 1.

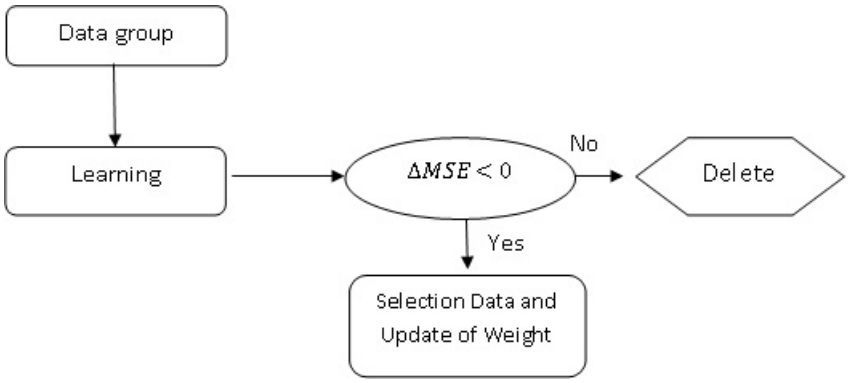


Fig. 1. Block Diagram of Integrating Data Selection and ELM

Note:
$\triangle\ MSE = MSE_{t+1} - MSE_t$

4. Experimental Settings and Dataset

4.1. Experimental Settings

This section discusses in more detail research procedure and dataset. This study is proposed new learning methods that able to do data selection and also classification. In this paper, we will be compared the proposed method with basic ELM. Procedure of research in this study will be presented in the flow chart of research as follows:

The first step in this research is pre-processing data, in this case used Z_{score} normalization. The formulation Z_{score} as follow:

$$Z_{score} = \frac{x_i - \bar{x}}{\sigma(x)} \tag{10}$$

note:
$x_i = \{x_1, x_2, ..., x_n\} \in R$
$\bar{x}$ = mean of data
$\sigma(x)$ = standard deviation

After pre-processing data (normalization), then conducted training data use 5-fold cross validation. In other word, the training data in this case using 80% data from existing datasets. After the training process is complete, further testing which uses 20% of the data.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

This research will compare Integrating Data Selection and ELM with standard Extreme Learning Machine, Back-propagation(BPNN), and Support Vector Machine(SVM). They have been compared using thirteen classification dataset that is all binary classification.

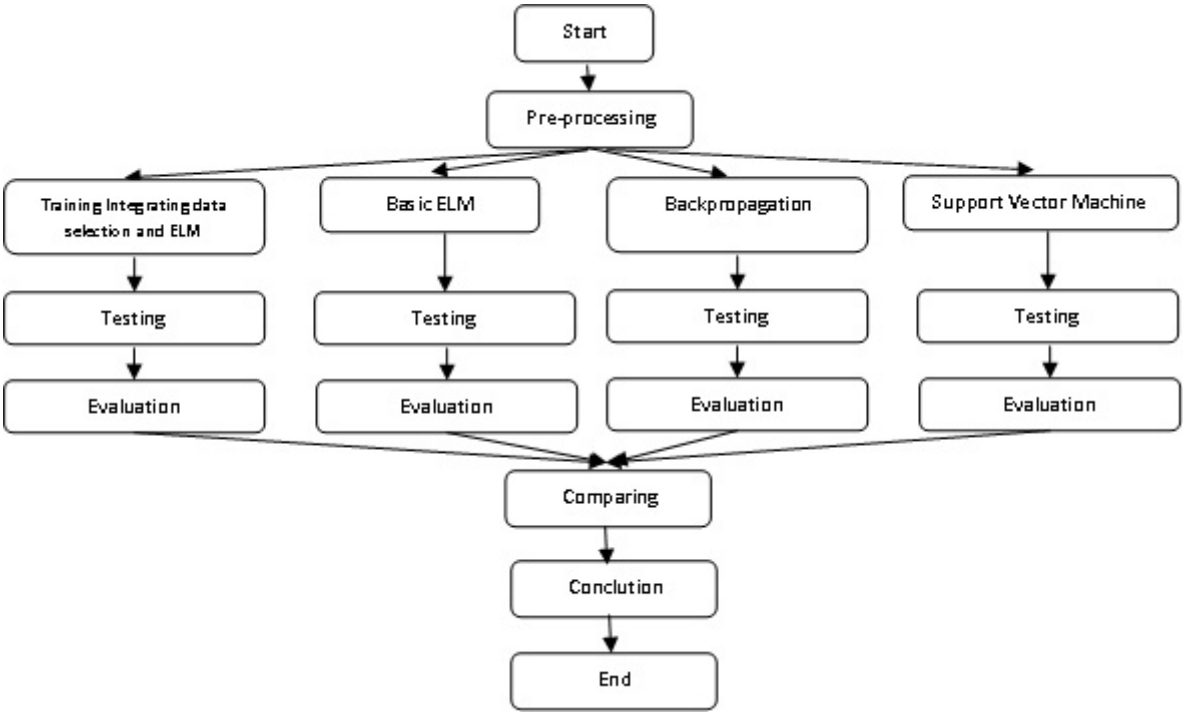


Fig. 2. Scheme of Methodology

Table 1. Table of Confusion Matrix for two class problem

		Actual	
		True	False
Prediction	True	TP (True Positive)	FP (False Positive)
	False	FN (False Negative)	TN (True Negative)

Accuracy, precision, recall, specificity, and G-Mean data were used as evaluation measure to compare those algorithms. Confusion matrix and evaluation measure formula as follow:

Accuracy is standard evaluation measurement that has been used in classification, but it is not suitable in imbalanced class classification. An imbalanced datasets, not only skewed of the class distribution, but the misclassification cost is often uneven too. The minority class example are often more important than the majority class examples. Precision is a measure of correctness, that out of positive labeled examples, how many are really a positive example[20]. Then, recall is a measure of completeness or accuracy of positive examples that how many examples of the positive class were labeled correctly[21]. Accuracy, precision and recall was defined:

$$Accuracy = \frac{TP + TN}{totalof\,sample} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

G*mean*(Geometric mean) is one of evaluation measure for imbalanced data. G*mean* is indicating balance performance between majority and minority class. To get G*mean* value, firstly we must get the value of sensitivity and specificity. Sensitivity is the accuracy of the positive sample data to measure the sensitivity same as the equation for measuring Recall. Whereas, specificity is the accuracy of the data sample is negative. G*mean* was defined:

$$Gmean = \sqrt{sensitivity \times specificity} \tag{15}$$

Specificity was defined:

$$Specificity = 1 - \frac{FP}{FP + TN} \tag{16}$$

4.2. Dataset

The datasets in this case is taken from the UCI Machine Learning Repository and Benchmark Data Sets for Highly Imbalanced Binary Classification (BDS)[22]. From the eleven selected dataset there are some attributes missing data, so that the pieces of data that have the lost attributes was deleted. That data description as follow:

Table 2. Table of dataset

Dataset	Sata size	Input feature	+ data	- data	Data From
QSAR biodegradation (QSAR)	1055	41	356	699	UCI
Spambase (Sp)	4601	51	1813	2788	UCI
Glass1 (G1)	214	10	70	144	UCI
Glass2 (G2)	214	10	76	138	UCI
Glass3 (G3)	214	10	17	197	UCI
Wilt (W)	4339	5	74	4265	UCI
Balance (B)	626	4	26	600	BDS
Abalone(A)	4175	8	210	3968	BDS
Yeast (Y)	1485	8	33	1452	BDS
Solar flare (Sf)	1390	10	44	1346	BDS
Mammographi (M)	11183	6	260	10923	BDS
Forest Cover (FC)	2267	12	189	2078	BDS
Letter Img (LI)	20000	16	734	19266	BDS

Glass data is conditioned to binary classification. Glass1 is data for building-windows-float-processed and not building-windows-float-processed , Glass2 for vehicle-windows-float-processed, and Glass3 is containers. All simulation in this paper have been carried out in MATLAB 2012b environment running in an AMD E-350 Processor 1,60GHz.

5. Experimental Results and Discussions

In this section we will discuss the results of this research, that is a comparison of the performance Integrating Data Selection and ELM and standard Extreme Learning Machine. The performance measure in this case including accuracy, precision, recall, specificity, and G*mean*.

5.1. Accuracy and Precision

The result of accuracy is shown in the table 3. From that table we can show that accuracy of IDELM not always better than the ELM, BPNN and SVM, because accuracy is viewed from the correct classification only, without seeing the balanced data or not. From diagram, we know the best accuration in Wilt, Spambase, Glass1, Yeat,Abalone, Solar flare, Mammography, Forest cover, and Letter Img data is IDELM, that is 0.99, 1, 1,0.98, 0.95, 0.97, 0.99, 0.92, and 0.98 consecutive. If we calculate that average of accuracy from all data the best accuracy is BPNN. But in this case we cannot see the performance from accuracy only, because classical evaluation measure (general accuracy) has no sense when evaluating the performances of a classifier over imbalanced domains[16].

Table 3. Table of Accuracy, Precision and recall

Data	Accuracy				Precision				Recall			
	ELM	IDELM	BP	SVM	ELM	IDELM	BP	SVM	ELM	IDELM	BP	SVM
W	0.98	0.99	1.00	0.99	0	0.99	0.96	0.57	0	1	0.76	0.57
Sp	0.86	1.00	0.93	0.83	0.84	1.00	0.92	0.96	0.79	1	0.90	0.96
qsar	0.80	0.83	0.86	0.83	0.74	0.70	0.81	0.84	0.63	1	0.77	0.84
B	0.96	0.95	0.96	0.56	0	0.85	0	0.08	0	1	0	0.08
G1	0.87	1.00	0.99	0.98	0.87	1.00	0.97	1.00	0.74	1	0.99	1.
G2	0.69	0.93	0.94	0.89	0.50	0.89	0.89	0.85	0.60	1	0.95	0.85
G3	0.62	0.95	0.93	0.97	0.15	0.93	0.68	0.87	0.73	1	0.78	0.87
Y	0.59	0.98	0.98	0.81	0.02	0.97	0.00	0.10	0.40	1	0	0.10
A	0.63	0.95	0.95	0.80	0.10	0.78	0	0.19	0.80	1	0	0.19
Sf	0.51	0.97	0.97	0.68	0.04	0.93	0	0.08	0.76	1	0	0.08
M	0.99	0.99	0.99	0.85	0	0.99	0.89	0.07	0	1	0.12	0.87
FC	0.92	0.92	0.92	0.85	0	0.86	0.61	0.34	0	1	0.18	0.92
LI	0.98	0.99	0.98	0.9	0	0.84	0.83	0.15	0	1	0.1	0.92
Average	0.8	0.9576	0.9538	0.8415	0.2507	0.9023	0.5815	0.4692	0.4192	1	0.4269	0.6346

From above table we can also see that almost all data, IDELM has the best precision. That means, IDELM can classify positive data better than ELM, BPNN, and SVM. In some Data ELM and BPPN cannot classify positive data correctly. So, that precision result is 0 because value of TP and FP is zero, as Wilt, Balance, Abalone, and Solar flare data.

5.2. G-mean

The table of specificity and G-mean is shown in the table 4. In table 4, We can show that almost in all data, IDELM have the best G-Mean. That mean, performance IDELM better than ELM, BPNN,and SVM for the case of imbalanced data. It is very visible in the data that have strong imbalanced data, that is on the data Wilt, Spambase, Glass1, Yeat,Abalone, Solar flare, Mammography, Forest cover, and Letter Img. For data Balance,Yeast,Abalone and Solar Flare , BPNN have G-Mean value of 0 because sensitivity is 0, that mean BPNN can not classify the positive data correctly.

Table 4. Table of Specificity and G-Mean

Data	Specificity				G-Mean			
	ELM	IDELM	BP	SVM	ELM	IDELM	BP	SVM
W	1	0.92	0.999	0.991	0	0.96	0.871	0.754
Sp	0.633	0.993	0.945	0.988	0.706	0.997	0.924	0.976
Qsar	0.907	0.85	0.918	0.939	0.755	0.92	0.842	0.890
B	0.925	0.95	1	0.522	0	0.98	0	0.203
G1	0.73424	1	0.93	0.993	0.739	1	0.957	0.997
G2	0.492	0.9	0.9	0.942	0.543	0.944	0.923	0.894
G3	0.514	0.91	0.984	0.9	0.614	0.95	0.875	0.883
Y	0.701	0.91	0.998	0.801	0.529	0.95	0	0.277
A	0.601	0.93	0.999	0.794	0.693	0.97	0	0.389
Sf	0.634	0.94	1	0.589	0.692	0.97	0	0.222
M	0.7	0.98	0.89	0.88	0.7	0.99	0.344	0.92
FC	1	0.85	0.99	0.82	0	0.92	0.42	0.87
LI	1	0.98	0.99	0.9	0	0.99	0.3	0.91
Average	0.7570	0.9317	0.9725	0.8583	0.4593	0.9649	0.4981	0.7065

5.3. CPU Time

First, we discuss about the result of CPU time, the result of CPU time shown in the following table:

Table 5. Table of CPU Time

Data	ELM(s)	IDELM(s)	BP(s)	SVM(s)
Wilt	0.0281	5.2947	23.2067	3.2448
Spambase	0.2714	4.829781	486.2551	4.8634
Qsar	0.0406	1.198	25.3065	1.10767
Balance	0.0187	0.911	2.6458	0.4930
Glass1	0.0094	0.2028	3.4195	0.1997
Glass2	0.0094	0.218	3.6099	0.1966
Glass3	0.0218	0.3762	3.2823	0.1903
Yeast	0.0218	1.95	10.8889	0.7363
Abalone	0.0281	7.279	38.8973	3.6878
Solar Flare	0.0125	1.797	7.4038	0.7582
Mammography	0.109	7.179	49.1247	5.257
Forest Cover	0.078	2.719	30.1706	1.7784
Letter Img	0.1903	87.2435	280.9079	69.4828
Average	0.0591	9.3229	74.2399	7.0766

From above table we can show that the faster CPU time is standart ELM, second is SVM, then IDELM. Backpropagation is slower than other algorithms. Spambase data have slowest cpu time, because that data have many attribute and element. That have 51 attribute and 4601 data. Because the cpu time not only depend on the type of algorithm, but it also depend on a lot of data and attribute of data.

6. Conclusion

This paper discusses and compares four classification model and thirteen data that have imbalanced data. All the four investigated models offer comparable classification accuracies. ELM has a good average of CPU time in almost all data, that is 0.0591 second, whereas CPU Time of SVM, IDELM, and BPNN is 7.0766, 9.3229, and 74.2399 second respectively. The best average of accuracy is IDELM, that is 95.76%.Performance for imbalanced data classification problem can not be measured with the conventional accuracy, so in this study using precision, sensitivity, specificity, and G Mean. From the above description of some methods that have the best average precision, recall, and G-mean is IDELM, that is 90.23%,100%, and 96.49%, while average of ELM, BPNN, and SVM precision is 25.07%, 58.15%, and 46.92%. Average of ELM, BPNN, and SVM recall is 41.92%, 42.69%, and 63.46%. then average of ELM, BPNN, and SVM G*mean* is 45.93%, 49.81%, and 70.65%. G*mean* can be used to general measure of accuracy in imbalanced data problem. So, from the result of G*mean*, we can conclude the best method is IDELM.

References

1. Tom M. Mitchell, Machine Learning, McGraw-Hill, 1997.
2. Umi Mahdiyah, M.I Irawan, dan E.M Imah. Study Comparison Backpropogation, Support Vector Machine, and Extreme Learning Machine for Bioinformatics Data. Jurnal Ilmu Komputer dan Informasi. 2015; 50(1): 55-62.
3. M.I Irawan, Siti Amiroch."Construction of Phylogenetic Tree Using Neighbor Joining Algorithms to Identify The Host and The Spreading of SARS Epidemic" Journal of Theoretical and Applied Information Technology. 2015; 71(3): 424-429.
4. Guang-Bin Huang, Zhu, Qin-Yu and Chee-Kheong Siew. Extreme learning machine: Theory and applications, Neurocomputing. 2006; 70:489-501.
5. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme Learning Machine: A New Learning Scheme of FeedForward Neural Network. Neurocomputing. 2004; 40: 7803-8359.
6. Xiaozhuo Luo, F. Liu, Shuyuan Yang, Xiaodong Wang, Zhiguo Zhou. Joint sparse regularization based Sparse Semi-Supervised Extreme Learning Machine (S3ELM) for classification. Neurocomputing. 2015; 73: 149-160.

7. Hong-Gui Han, Li-Dan Wang, Jun-Fei Qiao. Hierarchical extreme learning machine for feedforward neural network. Neurocomputing. 2014; 128: 128-135.
8. Guang-Bin Huang. An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels. 2014; 376-390.
9. Junchang Xin, Zhiqiong Wang, Luxuan Qu, Guoren Wang. Elastic extreme learning machine for big data classification. Neurocomputing. 149:464-471
10. Wentao Maoa, Shengjie Zhao, Xiaoxia Mu. Haicheng Wang. Multi-dimensional extreme learning machine. Neurocomputing. 2015; 146: 160-170.
11. H. He, E.A. Garcia. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 2009; 21 (9) : 12631284.
12. Weiwei Zong, Guang-Bin Huang, Yiqiang Chen. Weighted extreme learning machine for imbalance learning. Neurocomputing. 2013;101:229-242
13. Mohamed Bekkar and Taklit A. Imbalanced Data Learning Approaches. International Journal of Data Mining and Knowledge Management Process (IJDKP). 2013; 3(4):15-33.
14. C.Y. Lee, M. R. Yang, L. Y. Chang, Z. J. Lee. A Hybrid Algorithm Applied to Classify Unbalanced Data. Proceeding of International Conference on Networked Computing and Advanced Information Management. 2010: 618 621.
15. E.M. Imah, W. Jatmiko, T. Basarudin. Adaptive Multilayer Generalized Learning Vector Quantization (AMGLVQ) as new algorithm with integrating feature extraction and classification for Arrhythmia heartbeats classification. Systems, Man, and Cybernetics (SMC). 2012; 150-155.
16. E.M. Imah, W. Jatmiko, T. Basarudin.Electrocardiogram for Biometrics by using Adaptive Multilayer Generalized Learning Vector Quantization (AMGLVQ): Integrating Feature Extraction and Classification. 2013;5(6): 1891- 1917.
17. K.K. Paliwal, M. Bacchiani, and Y. Sagisaka.. Simultaneous design of feature extractor and pattern classifier using the minimum classification error training algorithm. Neural Networks for Signal Processing . V. Proceedings of the 1995 IEEE Workshop. 1995: 6776.
18. S. Chen and H. He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evolving Systems. 2010;2(1):35-50
19. Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew. Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes. Neural Network. 2006;17(4):(879-892)
20. J. Weng, Cheng G, Poon, A New Evaluation Measure for Imbalanced Datasets, in Seventh Austraasian Data Mining Conference (AusDM 2008), 2008.
21. Mohamed Bekkar, Dr.Hassiba Kheliouane Djemaa, Dr.Taklit Akrouf Alitouche. Evaluation Measures for Models Assessment over Imbalanced Data Sets. Journal of Information Engineering and Applications. 2013; 3(10):27-38
22. Ding, Zejin, "Diversified Ensemble Classifiers for Highly Imbalanced Data Learning and their Application in Bioinformatics." Dissertation, Georgia State University, 2011.

STUDY COMPARISON BACKPROPOGATION, SUPPORT VECTOR MACHINE, AND EXTREME LEARNING MACHINE FOR BIOINFORMATICS DATA

Umi Mahdiyah[1], M. Isa Irawan[1], and Elly Matul Imah[2]

[1]Faculty of Mathematics and Science, Institut Teknologi Sepuluh Nopember, Keputih, Surabaya, 60111, Indonesia
[2]Universitas Negeri Surabaya, Jl. Ketintang, Surabaya, 50231, Indonesia

E-mail: umi13@mhs.matematika.its.ac.id

Abstract

A successful understanding on how to make computers learn would open up many new uses of computers and new levels of competence and customization. A detailed understanding on information- processing algorithms for machine learning might lead to a better understanding of human learning abilities and disabilities. There are many type of machine learning that we know, which includes Backpropagation (BP), Extreme Learning Machine (ELM), and Support Vector Machine (SVM). This research uses five data that have several characteristics. The result of this research is all the three investigated models offer comparable classification accuracies. This research has three type conclusions, the best performance in accuracy is BP, the best performance in stability is SVM and the best performance in CPU time is ELM for bioinformatics data.

Keywords: *Machine Learning, Backpropagation, Extreme Learning Machine, Support Vector Machine, Bioinformatics*

Abstrak

Keberhasilan pemahaman tentang bagaimana membuat komputer belajar akan membuka banyak manfaat baru dari komputer. Sebuah pemahaman yang rinci tentang algoritma pengolahan informasi untuk pembelajaran mesin dapat membuat pemahaman yang sebaik kemampuan belajar manusia. Banyak jenis pembelajaran mesin yang kita tahu, beberapa diantaranya adalah *Backpropagation* (BP), *Extreme Learning Machine* (ELM), dan *Support Vector Machine* (SVM). Penelitian ini menggunakan lima data yang memiliki beberapa karakteristik. Hasil penelitian ini, dari ketiga model yang diamati memberikan akurasi klasifikasi yang sebanding. Penelitian ini memiliki tiga kesimpulan, yang terbaik dalam akurasi adalah BP, yang terbaik dalam stabilitas adalah SVM dan CPU time terbaik adalah ELM untuk data bioinformatika.

Kata Kunci: *Machine Learning, Backpropagation, Extreme Learning Machine, Support Vector Machine, Bioinformatika*

1. Introduction

A successful understanding of how to make computers learn would open up many new uses of computers and new levels of competence and customization. And a detailed understanding of information- processing algorithms for machine learning might lead to a better understanding of human learning abilities (and disabilities) as well [1]. Many type of machine learning that we know, some of them are Backpropagation (BP), Extreme Learning Machine (ELM), and Support Vector Machine (SVM).

First Machine Learning is backpropagation. Backpropagation was initially formulated by Webros in 1974, which was later modified by Rumelhart and McClelland [2]. Backpropagation is the gradient descent type algorithm, which has connection parameter for each step or iteration. But, this algorithm can provide harmony result between "network capability" to recognize the patterns which used for training and "network capability" to respond correctly to the input patterns that similar (but not equal) to the training pattern. Then, Backpropagation algorithm has limit classification accuracy, because if the output value is different from the target value, an error will be calculated, and then taken from the output layer to the input layer (Backpropagation process).

Until now, many researchers have developed and implemented the BP algorithm [3-7]. Implementation BP for classification was implemented in most problem as bioinformatic, biomedical, chemistry, art, environment, etc. Several research use this algorithm because BP

has minimum error and effective for some problem, especially in this study for classification problem.

The next powerful machine learning is Support Vector Machines (SVM). Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. This Machine learning is different from Machine Learning in ANN. In SVM when training process, SVM isn't training all data, but this algorithm just training support vector data. Based idea from this algorithm is optimization margin hyper plane. Although SVM need big memory and need long time to process data, but many researcher use SVM to solve some problem [8-11], because this machine learning have high performance.

Extreme Learning Machine is one of a new learning algorithm in neural networks, which has the Single-hidden Layer feed-forward Network (SLFN). ELM has a very fast learning capability and training small error [12]. First ELM was introduced by Huang in 2004 as Single-hidden Layer Feed-forward Network. ELM was made to overcome the weaknesses of the feed-forward neural networks problem that learning speed. Traditionally, feed-forward neural network using gradient-based learning algorithm for training, as well as all the parameters (input weight and hidden bias) are determined by iterative network, to solve that problem, Extreme Learning Machine using minimum norm least-squares (LS) solution of SLFNs.

Unlike the traditional function approximation theories which require to adjusted input weights and hidden layer biases, input weights and hidden layer biases can be randomly assigned if only the activation function is infinitely differentiable [13]. So, ELM can be faster than prior the neural network algorithm previously.

ELM has been applied and developed in various fields [14-18]. Up until now this algorithm is developed, the main reason many research use this algorithm because this algorithm is simple and faster than several algorithm in ANN.

In this paper, we will compare three algorithm for classification several bioinformatics data that have some criteria. Dataset was taken from UCI Machine Learning. We use general BP, SVM using linear kernel and general ELM. We choose Backpropagation (BP), Extreme Learning Machine (ELM), and Support Vector Machine (SVM) because that all machine learning have good performance for classification [2-18].

The rest of the paper is organized as follows. In Section 2, we present detail of BP, SVM, and ELM. In Section 3, we describe dataset and experimental settings. Following that, Section 4 provides experimental results and discussions. Finally, we draw conclusions in Section 5.

2. Related Work

This section will discuss detail the four categories of classifier.

2.1 Backpropagation

Backpropagation is one of many supervised machine learning. Backpropagation is the gradient descent type algorithm .This algorithm has two phases for processing data. First phase, input vector given to input layer, continued to hidden layer then finding output value in output layer. Second phase, if the output value is different from the target value, an error will be calculated, and then taken from the output layer to the input layer (Backpropagation process) [19].

In BP, transfer function must fulfill several conditions: continue differentiable, not descending function. In this paper we use sigmoid binary function:

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

$$f'(x) = f(x)\big(1 - f(x)\big) \tag{2}$$

Backpropagation training follow these step [20]:

a. Initialize neuron's weight with random number.

Forward propagation

b. Each input layer neuron receives the inputs and passes it to the connected hidden layer's neuron.
c. Compute output value from hidden layer z_j, $j = 1,2 \dots, n$, with the weight is v_{ji} that connected input layer and hidden layer.

$$z_{net_j} = v_{j0} + \sum_{i=1}^{p} x_i v_{ji} \tag{3}$$

$$z_j = f\left(z_{net_j}\right) = \frac{1}{1 + e^{-z_{net_j}}} \tag{4}$$

If the hidden layer end, we continue to output layer.

d. Compute all output values from output layer as equation(3) and equation(4).

$$y_{net_k} = w_{k0} + \sum_{i=1}^{p} x_i w_{ki} \quad (5)$$

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}} \quad (6)$$

Backpropagation

e. Compute the output layer's error factor based on the error in each of the output layer's neuron using this equation

$$\delta\_y_k = (t_k - y_k) f'(y_{net_k}) \quad (7)$$

$$\Delta w_{kj} = \propto \delta\_y_k z_j \quad (8)$$

f. Compute the hidden layer's error factor based on error in each of the hidden layer's neuron using this equation

$$\delta\_znet_j = \sum_{k=1}^{m} \delta_{y_k} w_{kj} \quad (9)$$

$$\delta\_z_j = \delta\_znet_j f'(\delta\_znet_j) \quad (10)$$

$$\Delta v_{ji} = \propto \delta\_z_j x_i \quad (11)$$

g. The last step count all change of weight of neurons.

$$w_{kj}(new) = w_{kj}(old) + \Delta w_{kj} \quad (12)$$

$$v_{ji}(new) = v_{ji}(old) + \Delta v_{ji} \quad (13)$$

This is architecture of Backpropagation Algorithm [19]:

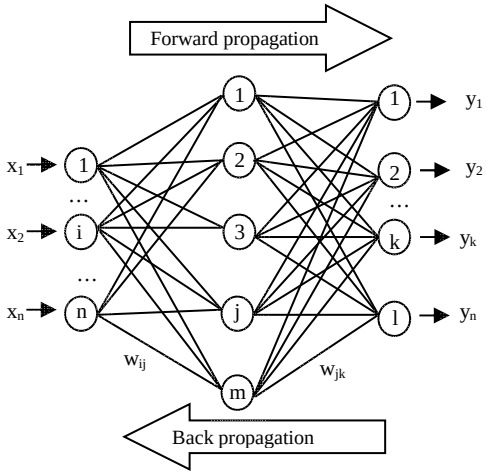


Figure 1. Backpropagation architecture.

This architecture has two processes, which are Forward propagation and back propagation. That is all was explained before.

2.2 Support Vector Machine

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. Originally, SVM is a training algorithm for linear classification. For non - linear case, SVM maps data sets of input space into a higher dimensional feature space, which is linear and the large - margin learning algorithm is then applied, the mapping can be done by kernel functions. Because, in the high dimensional feature space, that have maximal margin between the classes can be obtained, that called linear hyper plane classifiers [21].

This is step of support vector machine algorithm:

Given training data set D and q feature of each data, there are Φ, such that:

$$\Phi: D^q \rightarrow D^r \quad (14)$$

$$x \rightarrow \Phi(x) \quad (15)$$

r is new feature set that result from mapping D. while x is training data, which $x_1, x_2, \ldots x_n \in D^q$ is dataset that mapped to r dimension. The data that used to training is:

$$(\Phi(x_1), y_1, \ldots\ldots, \Phi(x_n), y_n) \in D^q \quad (16)$$

Mapping process in SVM needs dot product operation, which denoted $\Phi(x_i).\Phi(x_i)$. We can compute that dot product without knowing transform function of Φ . This computation technique is called kernel trick. Many type of kernel trick in SVM is there

TABLE I
KERNEL FUNCTION

Kernel	Function
Linear	$K(x,y) = x.y$
Polynomial	$K(x,y) = (x.y + v)^d$
Gaussian RBF	$K(x,y) = \exp\left(\frac{-\|x-y\|^2}{2.\sigma^2}\right)$
Sigmoid(tangent hyperbolic)	$K(x,y) = tanh(\sigma(x.y) + c)$
Multiquadratic Invers	$K(x,y) = \frac{1}{\sqrt{\|\|x-y\|\|^2 + c^2}}$

All explanation above can be resumed in illustration of hyperplane SVM. This is an illustration of margin hyperplane in SVM:

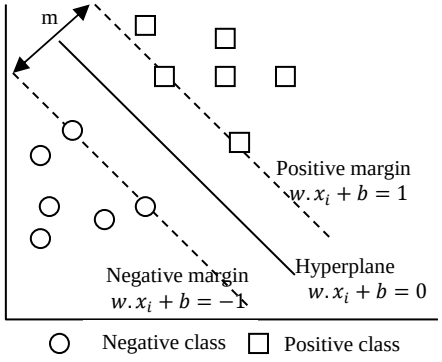


Figure 2. Margin hyperplane SVM.

SVM has positive margin, hyperplane and negative margin.

2.3 Extreme Learning Machine

Extreme learning machine is a group of supervised learning methods too as BP and SVM. This algorithm is one of a new learning algorithm in neural networks, which has the Single-hidden Layer feed-forward Network (SLFN). ELM has a simple algorithm, very fast learning capability and training small error [12]. First ELM was introduced by Huang in 2004 as Single-hidden Layer Feed-forward Network (SLFNs). ELM made to overcome the weaknesses of the feed-forward neural networks problem that learning speed. Traditionally, feed-forward neural network using gradient-based learning algorithm for training, as well as all the parameters (input weight and hidden bias) are determined by iterative network, to solve that problem, Extreme Learning Machine using minimum norm least-squares (LS) solution of SLFNs.

A standard single layer feedforward neural network with n hidden neurons and activation function g(x) can be mathematically modeled as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i . x_j + b_i) = o_j, \qquad (17)$$

$$j \in [1, N] \qquad (18)$$

Which $w_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ is input weight. b_i is bias hidden layer, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{in}]^T$ output weight, and $w_i . x_j$ is inner product from w_i *and* x_j and o_j output from this algorithm.

Standard SLFN with $\tilde{N}$ hidden nodes and activation function g(x) can approximate these N samples with zero error means that if $\sum_{i=1}^{\tilde{N}} \|o_j - t_j\| = 0$, there exist β_i, w_i dan b_i such that [12]

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i . x_j + b_i) = t_j , \ j \in [1, N] \qquad (19)$$

That equation can be write as

$$H\beta_i = T \qquad (20)$$

Note:

H is output matrix hidden layer.

$$H = \begin{bmatrix} g(w_1 . x_1 + b_1) & \cdots & g(w_{\tilde{N}} . x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1 . x_N + b_1) & \cdots & g(w_{\tilde{N}} . x_N + b_{\tilde{N}}) \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}$$

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}$$

ELM algorithm is derived from the minimum norm least squares solution SLFNs. Although, ELM is "generalized" of SLFN but hidden layer (feature mapping) of the ELM does not need to be tuned. Main concepts of ELM as presented in the journal Huang (2006), as follow there:

Given training set

$$\aleph = \{(x_i, t_i) | x_i \in \boldsymbol{R^n}, t_i \in \boldsymbol{R^m}, \ i \in [1, N]\},$$

activation function $g(x)$, and the number of hidden nodes $\tilde{N}$

Step 1: Insert random weights and biases w_i and b_i, $i \in [1, N]$

Step 2: Calculate the hidden layer output matrix H

Step 3: Calculate the output weights β

$\beta = H^{\dagger} T$

Which $T = [t_1, \dots \dots, t_N]^T$

$H^{\dagger}$ is *Moore–Penrose Generalized Inverse*

Theorem 2.1 A matrix $A \in M_{n \times m}(\mathbb{C})$. Exist *unique* $A^{\dagger} \in M_{m \times n}(\mathbb{C})$, that called *Moore–Penrose Generalized Inverse* if

1. $AA^{\dagger}A = A$;
2. $A^{\dagger}AA^{\dagger} = A^{\dagger}$
3. $AA^{\dagger} \in H_n$

4. $A^{\dagger}A \in H_m$

Then Activation function in the ELM must be infinitely differential (i.e sigmoid function, RBF, sine, cosine, exponential, etc.). Many hidden node depend on the number of training samples is N ($\tilde{N} \leq N$).

Several methods can be used to compute the Moore-Penrose Generalized Inverse of H, which are orthogonal projection, orthogonalization method, iterative method, and singular value decomposition (SVD).

3. Dataset and Experimental Settings

This section discusses in more detail research procedure and dataset. Procedure of research in this study will be presented in the flow chart of research as follows:

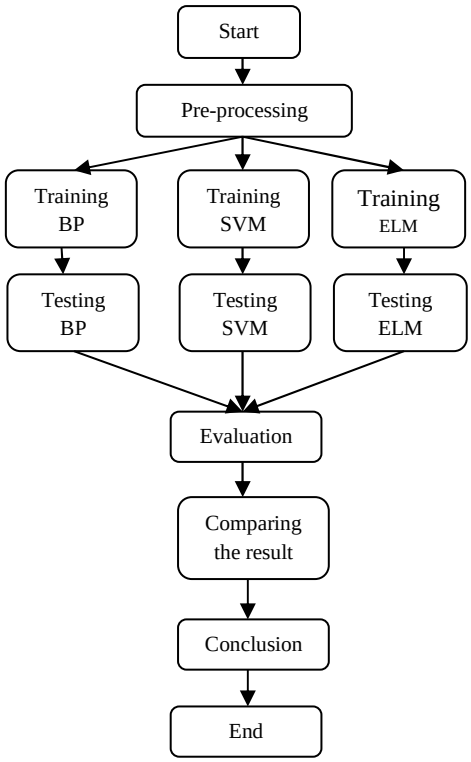


Figure 3. Scheme of methodology.

The first step in this research is preprocessing data, in this case used Z_{score} normalization. The formulation Z_{score} as follow:

$$Z_{score} = \frac{x_i - \bar{x}}{\sigma(x)} \tag{21}$$

note :

x_i $= \{x_1, x_2, \dots \dots x_n\} \in R$

$\bar{x}$ = mean of data

$\sigma(x)$ = standard deviations

After pre-processing data (normalization) then conducted training data, the training data in this case using 80% data from existing datasets. After the training process is complete, further testing which uses 20% of the data.

In this case is used sigmoid activation function. Threshold in this research use 0.5.

$$f(x) = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}} \tag{22}$$

This research will compare Backpropagation algorithm, Support Vector Machine using linear kernel, and standard Extreme Learning Machine. They have been compared using six classification dataset, that is all binary classification.

The datasets in this case is taken from the UCI Machine Learning Repository. From the six selected dataset there are some attributes missing data, so that the pieces of data that have the lost attributes was deleted.

Accuracy, precision, recall, CPU time, and misclassification data were used as evaluation measure to compare that three algorithm. Because the dataset uses not only balance data but also use imbalanced data.

TABLE II
TABLE OF CONFUSION MATRIX

		Actual	
		True	False
Prediction	True	TP *(True Positive)*	FP *(False Positive)*
	False	FN *(False Negative)*	TN *(True Negative)*

That formulation or accuracy, precision and recall as follow:

$$Accuracy = \frac{TP+TN}{total\ of\ sample} \tag{23}$$

$$Precision = \frac{TP}{TP+FP} \tag{24}$$

$$recall = \frac{TP}{TP+FN} \tag{25}$$

The dataset has been split into two categories based on data volume [21] and balancing data [22].

All simulation have been carried out in MATLAB 2012b environment running in an AMD E-350 Processor 1,60GHz.

TABLE III
DATA SET SUMMARY FOR CLASSIFICATION PERFORMANCE COMPARISON CATEGORIES BASED ON DATA VOLUME

Dataset	Data size	Input feature	+ data	- data	Data type
Small data					
Breast cancer	683	3	444	239	ID
Parkison	195	23	147	48	ID
Pima Indians Diabetes (PID)	768	8	268	500	ID
Promoter	106	58	53	53	BD
Big data					
QSAR biodegradation (QSAR)	1055	41	356	699	ID

*ID = imbalanced data
BD= balanced data

4. Experimental Results and Discussions

In this section we will discuss the results of this research, that is a comparison of the performance of BP, SVM, and ELM, including accuracy, precison, recall, cpu time and misclassification.

First, we analyze the result of accuracy, the result of accuracy shown in the following diagram:

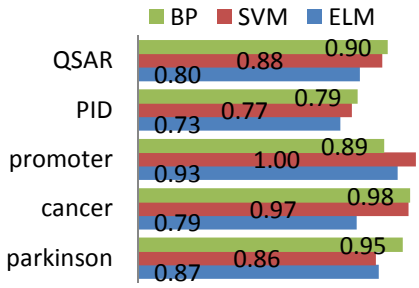


Figure 4. Diagram of accuracy.

From above data we can show the accuracy of ELM and SVM are not dependent on a small or big data but on balancing the data, whereas BP algorithm always achieves the highest accuracy. But in precision data SVM and ELM have precision higher than BP, that is accuracy for SVM and ELM is 1 and 0.93 then BP is 0.89. If we calculate accuracy of BP, SVM, and ELM for all data is $0.90 \pm 0.072, 0.90 \pm 0.093, 0.82 \pm 0.079$ in succession.

Second, we analyze the precision of each algorithm and each data.

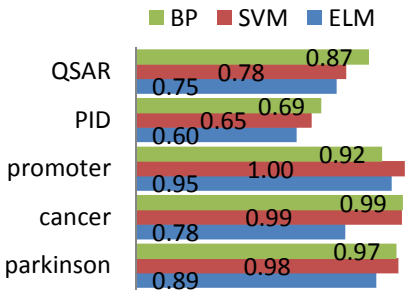


Figure 5. Diagram of precision.

From above figure we can show that precision is high if the data is balanced data, because the system or machine learn negative and positive data is balance, that is high precision in promoter data, precision of BP, SVM and ELM is 0.92, 1, and 0.9, then 0.95, 0.90, and 0.94, then 1, 0.97, and 0.86. If we calculate precision of BP, SVM, and ELM for all data is $0.89 \pm 0.16, 0.88 \pm 0.16, 0.79 \pm 0.14$ in succession.

Then, we analyze recall. That result of recall as follow:

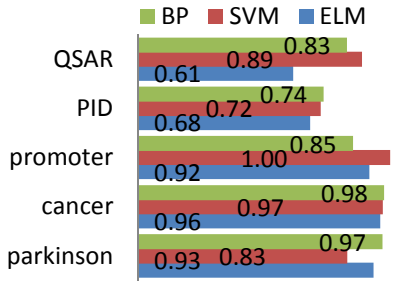


Figure 6. Diagram of recall.

Recall, accuracy, and precision in BP and ELM not only depend on data volume or balancing volume, but also dependent on determine of threshold, in this result we use 0.5 for all data. If we calculate recall of BP, SVM, and ELM for all data is $0.87 \pm 0.12, 0.88 \pm 0.11, and\ 0.82 \pm 0.16$ in succession.

We can show in three diagram above, accuracy, precision, and recall of BP in promoter dataset always above ELM and SVM. This occur phenomenon because SVM and ELM have best performance in balance dataset. Because promoter dataset have balanced characteristic, so ELM and SVM can surpass performance of BP.

The last, we show the diagram of CPU time and number misclassification data for each data and each algorithm. Result of CPU time and the number of misclassification data shown in the following diagram.

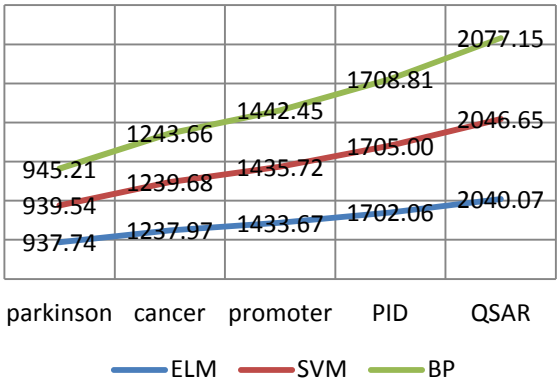


Figure 7. Diagram of CPU time.

From above diagram we can show that the CPU time of parkinson data using BP method is faster than CPU time in promoter data, however the parkinson data size is bigger than promoter data and the data type of promoter data is balanced data. That could be happened because attribute of promoter data is more than Parkinson data.

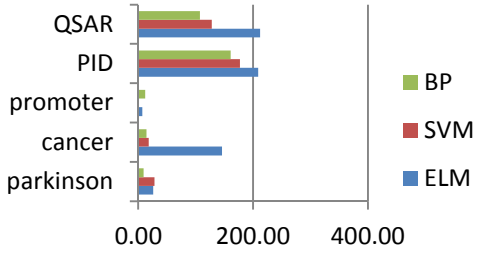


Figure 8. Diagram of the number of misclassification.

The result of CPU time is in seconds. From the above results it can be seen that the CPU time of the ELM and SVM is almost always the same as the average difference of them is 5 second, while the CPU time of the BP is always high, especially for large data and imbalance.

Then, the most of misclassification data is in the large data and imbalance data, because the system more learns a lot of majority data than the minority data. Not only characteristic of data but also determine of threshold also have influence of misclassification data.

5. Conclusion

This paper discusses and compares three classification model and seven data that have two characteristic. All the four investigated models offer comparable classification accuracies. ELM has a good performance in balance data, so if we use ELM for imbalanced data must be conditioned so that the data used balance, using undersampling or other. SVM has a very good performance but requires considerable memory in the training process. BP always has the best perform in accuracy, precision, and recall, the performance has a weakness at the time of computation. So we have three type conclusion, the best perform of accuracy is BP, the best perform of stability is SVM and the best perform of CPU time is ELM.

References

[1] T.M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[2] M.K. Alsmadi, K.B. Omar, S.A. Noah, & I. Almarashdah, "Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks", *IEEE International Advance Computing Conference*, pp. 296-299, 2009.

[3] K. Rudd, G.D. Muro, & S. Ferrari, "A Constrained Backpropagation Approach for the Adaptive Solution of Partial Differential Equations", *Neural Networks and Learning Systems*, vol. 25, pp. 571-584, 2014.

[4] A.H. Kridalaksana, E. Arriyanti, M.I.Ukkas, "Recognition of a human behavior pattern in paper rock scissor game using backpropagation artificial neural network method", *Information and Communication Technology (ICoICT)*, pp. 238 – 243, 2014.

[5] A.O. Ibrahim, S.M. Shamsuddin, N.B. Ahmad, and M.N. Salleh, "Hybrid NSGA-II of Three-Term Backpropagation network for multiclass classification problems "*Computer and Information Sciences (ICCOINS)*, pp. 1-6, 2014.

[6] N. Suciati, W.A. Pratomo, D. Purwitasari, "Batik Motif Classification Using Color-Texture-Based Feature Extraction and Backpropagation Neural Network", *Advanced Applied Informatics*, pp. 517 – 521, 2014.

[7] J.S. Bhalla, A. Aggarwal, "A Novel Method for Medical Disease Diagnosis Using Artificial Neural Networks Based On Backpropagation Algorithm", *The Next Generation Information Technology Summit*, pp. 55-61, 2013.

[8] F.C. Er, G.H. Hatay, E. Okeer, M. Yildirim, "Classification of phosphorus magnetic resonance spectroscopic imaging of brain tumors using support vector machine and logistic regression at 3T", *Engineering in Medicine and Biology Society*, pp. 2392 - 2395 , 2014.

[9] Y. Shuang, K.K. Tan, B. L. Sng, L. Shengjin, "Feature Extraction and Classification for Ultrasound Images of Lumbar Spine with Support Vector Machine", *Engineering in Medicine and Biology Society*, pp. 4659 – 4662, 2014.

[10] J. Manikandan & V.K. Agrawal, "Digital Circuit Design Using Support Vector Machines", *Communication Systems and Network Technologies*, pp. 1114 – 1118, 2014.

[11] G.Y. Wong, F.H.F. Leung, & S.H. Ling, "Predicting Protein-Ligand Binding Site Using Support Vector Machine with Protein Properties", *Transactions on Computational Biology and Bioinformatics*, vol. 10, No. 10, pp. 1517-1529, 2013.

[12] G.B. Huang, Q.Y. Zhu, & C.K. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[13] G.B. Huang, Q.Y. Zhu, & C.K. Siew, "Extreme Learning Machine: A New Learning Scheme of FeedForward Neural Network", *Neurocomputing*, vol. 40, pp. 7803-8359, 2004.

[14] X. Luo, F. Liu, S. Yang, X. Wang, & Z. Zhou, "Joint sparse regularization based Sparse Semi-Supervised Extreme Learning Machine (S3ELM) for classification", *Neurocomputing*, vol. 73, pp. 149-160, 2015.

[15] H.G. Han, L.D. Wang, & J.F. Qiao, "Hierarchical extreme learning machine for feedforward neural network", *Neurocomputing*, vol 128, pp. 128-135, 2014.

[16] G.B. Huang, "An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels", pp. 376-390, 2014.

[17] J. Xin, Z. Wang, L. Qu, & G. Wang, "Elastic extreme learning machine for big data classification", *Neurocomputing*, vol. 149, pp. 464-471, 2015.

[18] W. Maoa, S. Zhao, X. Mu, & H. Wang, "Multi-dimensional extreme learning machine", *Neurocomputing*, vol. 146, pp. 160-170, 2015.

[19] E. Prasetyo, *Data Mining: Konsep dan Aplikasi Menggunakan MATLAB*, Andi, Yogyakarta, p. 88, 2012.

[20] J.J. Siang, *Jaringan Saraf Tiruan dan Pemrogramannya Menggunakan MATLAB*. Andi Offset, Yogyakarta, p. 102-103, 2006.

[21] Yi Wang, Yi Li, Momiao Xiong, Li Jin, "Random Bits Regression: a Strong General Predictor for Big Data", *Chinese Journal of Computer*, 2005.

[22] G. Menardi & N. Torelli, "Training and Assessing Classification rules with Imbalanced Data", *Data Mining and Knowledge Discovery*, vol. 28, pp. 92-122, 2014.

Prosiding Seminar Nasional Matematika dan Pendidikan Matematika 2015
ISBN No. 978-979-028-728-0

PREDIKSI PROTEIN-LIGAN *BINDING SITE* MENGGUNAKAN *EXTREME LEARNING MACHINE* (ELM)

Umi Mahdiyah[1], M. Isa Irawan[2], Elly Matul Imah[3]

[1]Institut Teknologi Sepuluh Nopember, Keputih, Surabaya, 60111, Indonesia
[2]Institut Teknologi Sepuluh Nopember, Keputih, Surabaya, 60111, Indonesia
[3]Universitas Negeri Surabaya, Jl. Ketintang, Surabaya, 50231, Indonesia

Abstrak. Bioinformatika merupakan ilmu interdisipliner yang sudah banyak diaplikasikan dalam kehidupan sehari – hari. Salah satu aplikasi ilmu bioinformatika adalah pada desain obat berbantuan komputer. Pengembangan ilmu pengetahuan mengenai desain obat berbantuan komputer telah banyak dilakukan. Langkah awal desain obat berbantuan komputer adalah menemukan struktur 3D protein dan *binding site* ligandnya. *Binding site* merupakan kantong atau celah pada permukaan protein yang digunakan untuk mengikat *ligand* (obat). Prediksi situs pengikatan (*binding site*) dapat dirumuskan sebagai masalah klasifikasi biner. Dalam hal ini, salah satu algoritma jaringan syaraf tiruan akan digunakan untuk proses klasifikasi, yaitu *Extreme Learning Machine* (ELM). *Extreme Learning Machine* adalah salah satu algoritma klasifikasi, algoritma ini memiliki kecepatan belajar yang sangat baik. Sehingga, algoritma ini dapat dilakukan untuk memprediksi *ligand binding site* dengan menggunakan dua atribut, yaitu *score conservation*, dan interaksi potensial. Hasil dari penelitian ini menunjukkan nilai akurasi 76,05%, *precision* 78,01%, dan *recall* 60%.

***Kata kunci**: bioinformatika, desain obat, binding site, Extreme Learning Machine*

Pendahuluan

Bioinformatika merupakan ilmu interdisipliner yang melibatkan berbagai bidang ilmu, yang meliputi biologi molekuler, matematika, ilmu komputasi, kimia molekul, fisika, dll [1]. Selama ini, bioinformatika sudah banyak diaplikasikan pada dalam berbagai masalah, salah satunya adalah masalah desain obat. Konsep desain obat pada bioinformatika didasarkan pada fungsionalitas dari protein, yaitu pencarian sebuah senyawa untuk mengaktifkan atau menghambat fungsi protein target, sebab protein dapat mengekspresikan suatu informasi genetik. Seperti juga terdapat ribuan gen di dalam inti sel, masing-masing mencirikan sebuah sifat nyata dari organisme, di dalam sel terdapat ribuan jenis protein yang berbeda, masing-masing membawa fungsi spesifik yang ditentukan oleh gen yang sesuai.

Beberapa aplikasi bioinformatika dalam desain obat yang sudah banyak dikenal adalah *virtual screening*, *docking*, dan *de novo drug design*. Sebelum melakukan implementasi dengan aplikasi bioinformatika tersebut, struktur tiga dimensi dari target dan *binding site* protein-ligan harus ditemukan terlebih dahulu. *Binding site* adalah suatu lubang atau kantung yang terdapat pada permukaan protein yang nantinya akan digunakan untuk menambatkan suatu ligan (senyawa obat).

Prediksi dari *binding site* dapat dirumuskan sebagai masalah klasifikasi biner, yaitu untuk membedakan lokasi yang bisa mengikat suatu ligan dan lokasi yang tidak mengikat ligan [2]. *Extreme Learning Machine* merupakan salah satu *machine learning* untuk pengenalan pola dan klasifikasi dengan performansi yang bagus [3].

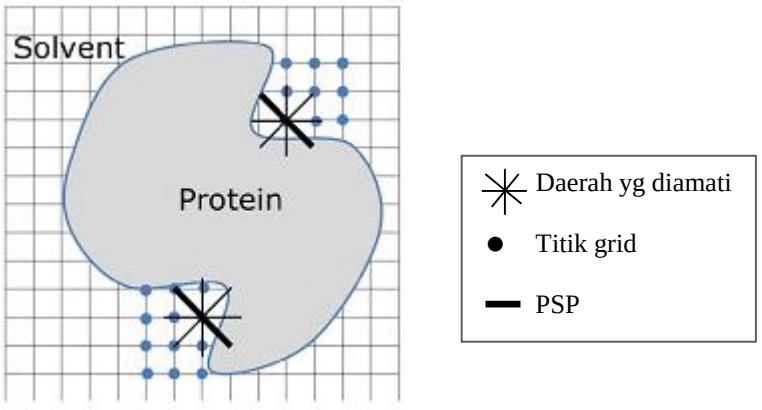
Teori Penunjang

2.4 Prediksi *Binding site* Protein-ligand

2.1.1 Prediksi dengan Pendekatan Berdasar Geometri

LIGSITE merupakan salah satu metode berbasis geometri untuk menemukan suatu *binding site*[4]. LIGSITE merupakan perbaikan dari algoritma yang dikembangkan oleh Levitt dan Banaszak yaitu program POCKET. Program ini dimulai dengan membuat Cartesian grid yang teratur. Kedua, pemeriksaan jarak diterapkan pada grid untuk memastikan atom protein tidak tumpang tindih dengan titik grid. Semua titik-titik grid, yang tidak tumpang tindih dengan atom protein, diberi label sebagai pelarut. Jika titik-titik grid luar protein yang tertutup oleh permukaan protein, yaitu titik-titik grid diapit oleh pasangan atom dalam protein, itu disebut *protein-solvent-protein (PSP) event*.

Langkah awal LIGSITE dan POCKET sama seperti yang dijelaskan sebelumnya. Keduanya sama – sama identifikasi *PSP event* berdasarkan koordinat atom. *Scan* LIGSITE untuk *pocket* sepanjang tiga sumbu dan empat diagonal ruang sementara POCKET hanya *scaning* pada tiga sumbu saja. Beberapa nilai akan digunakan untuk setiap titik grid merupakan jumlah PSP *event* yang terdeteksi saat scanning pada setiap sumbu. Artinya, semakin tinggi nilai titik grid, semakin besar kemungkinan titik grid akan menjadi kandidat *binding site*. Metode tersebut hanya berfokus pada karakteristik geometris dan tidak mempertimbangkan karakteristik lain dari protein [4]. Gambaran dari PSP *event* dapat dilihat pada gambar 2.1 [2].



Gambar 1 PSP event digunakan untuk mendeskripsikan fitur geometri sebuah titik grid

2.1.2 Prediksi dengan Pendekatan Berdasar Energi

PocketFinder adalah salah satu pendekatan berdasar energi untuk menentukan *binding site* ligan [5]. Langkah pertama dibuat grid 3D untuk *potential map*. Titik grid yang berpotensial (P) dihitung dengan menggunakan rumus Lennard-Jones:

$$P_P^0 = \sum_{i=1}^{N} \left(\frac{A_{XC}}{r_{pi}^{12}} - \frac{B_{XC}}{r_{pi}^{6}} \right)$$

r_{pi} merupakan jarak antara letak p (*grid point*) dan atom protein X_i. Parameter A_{XC} dan B_{XC} koefisien Van der Walls yang diambil dari *Empirical Conformational Energy Program for Peptides* (ECEPP).

ISBN No. 978-979-028-728-0

2.1.3 Prediksi dengan Pendekatan Berdasar *Sequence*

Semua residu dalam protein belum tentu selalu penting. Beberapa hal saja yang perlu untuk suatu struktur penting protein dan fungsi protein, sebaliknya sebagian yang lain dapat digantikan. Analisis konservasi adalah salah satu dari banyak metode yang digunakan untuk memprediksi fungsional residu penting dari *sequence* protein [6].

2.5 *Extreme Learning Machine*(ELM)

Extreme learning machine pertama kali diusulkan oleh Huang[7], yang merupakan algoritma *learning* sederhana untuk *Single Hidden Layer Feedforward Networks* (SLFN) dengan kecepatan *learning* dapat sangat cepat dibandingkan *algoritma learning feedforward network* tradisional.

2.1 *Single Hidden Layer Feedforward Networks* (SLFN)

Untuk sebuah himpunan N sampel yang berbeda(x_i, t_i) dengan $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$ dan $t_i = [t_{i1}, t_{i2}, \dots, t_{in}]^T \in \mathbb{R}^m$. Maka SLFN standar dengan $\tilde{N}$ hidden nodes dan fungsi aktivasi $g(x)$ dimodelkan dalam bentuk matematika sebagai berikut [7]:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i\left(x_j\right) = \sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i.x_j + b_i\right) = o_j$$
$$j \in [1, N]$$

Dengan $w_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ adalah vektor bobot input neuron ke- i dalam *hidden node*
b_i = bias *hidden layer*, dan
$\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{in}]^T$ merupakan bobot output
$w_i.x_j$ = *inner product* dari w_i dan x_j

SLFN standard dengan $\tilde{N}$ hidden nodes dan fungsi aktivasi $g(x)$ dapat mendekati N sampel dengan error rata – rata nol, $\sum_{i=1}^{\tilde{N}} \|o_j - t_j\| = 0$ terdapat β_i, w_i dan b_i sehingga

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i.x_j + b_i) = t_j\ ,\ \ j \in [1, N]$$

dapat ditulis,

$$H\beta_i = T$$

Dengan

$$H(w_1, \dots, w_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, x_1, \dots, x_{\tilde{N}}) = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_{\tilde{N}}.x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \cdots & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix} \text{ dan } T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}$$

Dengan H adalah output matrix hidden layer, kolom ke-i dari H adalah output hidden node ke-i.

2.2 Algoritma *Extreme Learning Machine*(ELM)

Algoritma ELM merupakan algoritma yang diperoleh dari solusi *minimum norm least square* SLFNs. ELM adalah "*generalized*" dari SLFN tetapi *hidden layer* (*feature maping*) dari ELM tidak perlu disetel. Berikut adalah konsep utama dari ELM [3]:
Diberikan *training set* $\aleph = \{(x_i, t_i) | x_i \in \boldsymbol{R^n}, t_i \in \boldsymbol{R^m},\ i \in [1, N]\}$, fungsi aktivasi $g(x)$, dan bilangan *hidden node* $\widetilde{N}$

Step 1: masukkan secara random bobot w_i dan bias b_i, $i \in [1, N]$
Step 2: hitung output matriks hidden layer H
Step 3: hitung bobot output β

$$\beta = H^{\dagger}T$$

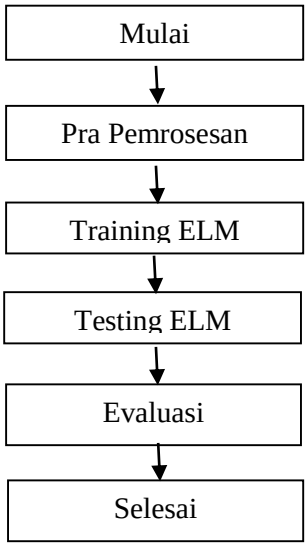
Dengan $T = [t_1, \dots\dots, t_N]^T$
$H^{\dagger}$ adalah *Generalized Inverse*

2.3 Metodologi

Pencarian *binding site* merupakan suatu langkah penting dalam desain obat berbantuan komputer. Dalam paper ini pencarian *binding site* diformulasikan dalam masalah klasifikasi biner. *Extreme Learning Machine* merupakan salah satu algoritma jaringan syaraf tiruan yang digunakan untuk klasifikasi. Dalam paper ini pencarian *binding site* akan dilakukan dengan menggunakan algoritma ELM, dengan mengaklasifikasikan *sequence* protein berdasarkan atribut yang digunakan.

Data protein pada penelitian ini diambil dari *webserver* LigAsite, yang masing – masing atributnya diperoleh dari *webserver* ConCavity [4]. Dalam penelitian ini digunakan 10 data protein untuk training dan 3 data protein untuk testing, dengan 2 atribut yang digunakan yaitu skor dari *sequence conservation* dan skor dari perhitungan interaksi potensial pada protein menggunakan *webserver* ConCavity.

Secara umum, tahapan – tahapan yang dilakukan dalam paper ini akan disampaikan pada diagram alir sebagai berikut:



Gambar 2. Diagram alir metodologi

ISBN No. 978-979-028-728-0

Berdasarkan diagram alir pada Gambar 1, maka secara lebih jelas akan dijelaskan lebih terperinci sebagai berikut:

a. Pra-Pemrosesan

Data yang diinputkan pada Extreme Learning Machine dinormalisasi terlebih dahulu sehingga mempunyai nilai dengan *range* tertentu. Dalam penelitian ini akan digunakan rumus normalisasi minmax [8]:

$$x = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1$$

b. Training ELM

Adapun langkah – langkah training ELM adalah sebagai berikut:

1) Masukkan secara random bobot w_i dan bias b_i, $i \in [1, N]$ dengan fungsi aktivasi g(x), yaitu fungsi akktivasi sigmoid [9]:

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

2) Hitung output matriks hidden layer H
3) Hitung bobot output β

$$\beta = H^{\dagger}T$$

c. Testing ELM

Berdasarkan bobot input dan bobot output yang didapatkan dari proses *training*, maka tahap selanjutnya adalah melakukan testing dengan ELM.

d. Evaluasi

Untuk membandingkan dan mengevaluasi metode yang diusulkan dengan metode yang lain digunakan perhitungan performa, dalam hal ini adalah akurasinya. Dalam penelitian ini diaplikasikan dua ukuran dalam metode dan dataset yang berbeda.

Untuk mengukur performa dari klasifikasi data pengujian *imbalance* dalam hal ini digunakan *confusion matrix, precision dan recall*. Berikut adalah *confusion matrix*[10]:

Tabel 3.1 Tabel *Confusion Matrix*

		Nilai Sebenarnya	
		True	False
Prediksi	True	TP *(True Positive)*	FP *(False Positive)*
	False	FN *(False Negative)*	TN *(True Negative)*

Rumus dari *precission* dan *recall* serta adalah sebagai berikut:

$$Accuracy$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

Dengan: TP = nilai *true positive*
FP = *false positive*
FN =*false negative*

Hasil Penelitian

Dalam penelitian ini digunakan 10 data protein untuk training dan 3 data protein untuk testing, dengan 2 atribut yang digunakan yaitu skor dari *sequence conservation* dan skor dari perhitungan interaksi potensial pada protein menggunakan *webserver* ConCavity.

Hasil dari penelitian ini disajikan dalam tabel berikut:

Tabel 1. Tabel Hasil Penelitian

Tipe	Hasil
Accuracy	76,05 %
Precision	78,01 %
Recall	60,00%
Waktu komputasi	5.881 detik

Hasil penelitian pada tabel di atas adalah persentase rata – rata akurasi, *precision* serta *recall* dari ketiga protein pada data testing. Dari tabel di atas dapat dilihat bahwa akurasi dari penerapan ELM untuk prediksi *binding site* dengan mengklasifikasikan *sequence* adalah 75,05%. Artinya, 75,05% asam amino dapat diklasifikasikan dengan benar.

Dapat dilihat pula bahwa *precision* dari penerapan ELM untuk prediksi *binding site* dengan klasifikasi *sequence* adalah 78,01%. Artinya, 78,01% asam amino yang berpotensi sebagai *binding site* dapat diklasifikasikan dengan benar.

Selanjutnya yang terakhir adalah *recall*. Dari tabel di atas dapat kita lihat bahwa *recall* dari penerapan ELM untuk prediksi *binding site* dengan klasifikasi *sequence* adalah 60%. Artinya, porsi dari prediksi data positif yang dikenali benar oleh algoritma adalah 60%. Dengan waktu komputasi yang dibutuhkan untuk proses testing selama 5.881 detik.

Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka kesimpulan dari penelitian ini adalah sebagai berikut:

a. Akurasi dari penerapan ELM untuk prediksi *binding site* dengan mengklasifikasikan *sequence* adalah 75,05%
b. *Precision* dari penerapan ELM untuk prediksi *binding site* dengan klasifikasi *sequence* adalah 78,01%.
c. *Recall* dari penerapan ELM untuk prediksi *binding site* dengan klasifikasi *sequence* adalah 60%.
d. Waktu komputasi yang dibutuhkan dalam penerapan ELM untuk prediksi *binding site* dengan klasifikasi *sequence* adalah 5.881 detik.

ISBN No. 978-979-028-728-0

Daftar Pustaka

[1] Shiyi N. Shen dan Jack A. Tuszynski, *Theory and Mathematical Methods for Bioformatics*, Springer, Verlag Berlin Heidelberg, 2008.

[2] Ginny Y. Wong, Frank H.F. Leung, dan S.H. Ling, "Predicting Protein-Ligand *Binding site* Using Support Vector Machine with Protein Properties", Transactions on Computational Biology and Bioinformatics, Vol. 10, No. 10, hal. 1517-1529, 2013.

[3] Guang-Bin Huang, Qin-Yu Zhu, dan Chee-Kheong Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, Vol. 70, hal. 489–501, 2006.

[4] Manfred Hendlich, Friedrich Rippmann dan Gerhard Barnickel, "LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins", *Journal of Molecular Graphics and Modelling*, Vol.15, hal. 359 –363, 1997.

[5] An, Jianghong, Totrov, Maxim dan Ruben Abagyan, "Pocketome via Comprehensive Identification and Classification of Ligand Binding Envelopes", *Molecular & Cellular Proteomics,Vol.* 4, hal. 752-761.

[6] John A. Capra, Roman A. Laskowski, Janet M. Thornton, Mona Singh , Thomas A. Funkhouser, "Predicting Protein Ligand Binding Sites by Combining Evolutionary *Sequence* Conservation and 3D Structure", *Computational Biology*, Vol. 5, hal.1-18, 2009.

[7] Guang-Bin Huang, Qin-Yu Zhu, dan Chee-Kheong Siew, "Extreme Learning Machine: A New Learning Scheme of FeedForward Neural Network", Neurocomputing, Vol. 40, hal. 7803-8359, 2004.

[8] Chengzhang Zhu, Jianping Yin, dan Qian Li, "A Stock Decision Support System Based on ELM", *Proceedings of the International Conference on Extreme Learning Machines (ELM2013)*, (eds) **Sun**, F., **Toh**, K.-A., **Romay**, M.G., **Mao**, K., Beijing, hal.67-79, 2013.

[9] Jong Jek Siang, 2009, *Jaringan Syaraf Tiruan dan Pemrogramanan Menggunakan MATLAB*, ANDI, Yogyakarta.

[10] Mohamed Bekkar, Dr.Hassiba Kheliouane Djemaa, Dr.Taklit Akrouf Alitouche, Evaluation Measures for Models Assessment over Imbalanced Data Sets, Journal of Information Engineering and Applications, Vol. 3, hal. 27-38, 2013.

BIOGRAFI PENULIS

Penulis bernama Umi Mahdiyah, lahir di Kediri, 29 September 1989, merupakan anak ketiga dari 4 bersaudara. Penulis menempuh pendidikan formal di TK Dharma Wanita (1994-1995), SDN Tenggerkidul 1 (1995-2001), MTs Hasan Muchyi Kapurejo (2001-2004), MAN 3 Kediri (2004-2007). Setelah lulus dari jenjang Madrasah Aliyah, penulis melanjutkan studi di FKIP Universitas Nusantara PGRI Kediri (2007-2011). Kemudian penulis mengikuti program Pra S2 Saintek angkatan pertama tahun 2012 Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember Surabaya. Kemudian penulis melanjutkan studi magister di Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember Surabaya. Penulis dapat dihubungi melalui e-mail: umimahdiyah@gmail.com.