TUGAS AKHIR – KI091391

IMPLEMENTASI METODE HYBRID JST-SOM PADA PREDIKSI *CHURN* PELANGGAN SELULER: STUDI KASUS PT. TELEKOMUNIKASI SELULER

NAUFAL AULIA RIZAL
NRP 5110 100 076

Dosen Pembimbing I
Arya Yudhi Wijaya, S.kom., M.kom.

Dosen Pembimbing II
Rully Soelaiman, S.kom., M.kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT – KI091391

IMPLEMENTATION HYBRID NN-SOM METHOD FOR PREDICTION CUSTOMER CHURN: STUDY CASE PT. TELEKOMUNIKASI SELULER

NAUFAL AULIA RIZAL
NRP 5110 100 076

Supervisor I
Arya Yudhi Wijaya, S.kom., M.kom.

Supervisor II
Rully Soelaiman, S.kom., M.kom

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA 2014

LEMBAR PENGESAHAN

IMPLEMENTASI METODE HYBRID JST-SOM PADA PREDIKSI *CHURN* PELANGGAN SELULER: STUDI KASUS PT. TELEKOMUNIKASI SELULER

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
NAUFAL AULIA RIZAL
NRP. 5110 100 076

Disetujui oleh Pembimbing Tugas Akhir:

1. Arya Yudhi Wijaya, S.Kom., M.Kom. ..... (Pembimbing 1)
   NIP:19840904 201012 1 002
2. Rully Soelaiman, S.Kom., M.Kom. ..... (Pembimbing 2)
   NIP:19700213 199402 1 001

SURABAYA
JANUARI, 2014

IMPLEMENTASI METODE HYBRID JST-SOM PADA PREDIKSI *CHURN* PELANGGAN SELULER: STUDI KASUS PT. TELEKOMUNIKASI SELULER

Nama : **Naufal Aulia Rizal**
NRP : **5110100076**
Jurusan : **Teknik Informatika – FTIf ITS**
Dosen Pembimbing I : **Arya Yudhi Wijaya, S.kom., M.kom.**
Dosen Pembimbing II : **Rully Soelaiman, S.Kom., M.Kom.**

Abstrak

Literatur marketing menyatakan bahwa lebih mahal biaya yang dikeluarkan provider untuk mendapatkan pelanggan baru daripada mempertahankan pelanggan loyal *yang sudah ada. Churn pelanggan adalah istilah yang digunakan pada industri telekomunikasi yang artinya adalah perpindahan pelanggan dari satu provider ke provider lainnya. Model prediksi churn pelanggan dikembangkan untuk mempertahankan pelanggan yang sudah ada. Manajemen churn sangat penting bagi aktivitas provider untuk mempertahankan pelanggan yang loyal. Sangat dibutuhkan metode untuk memprediksi churn dengan baik. Pasar industri telekomunikasi berkembang sangat kompetitif sehingga membuat manajemen churn pelanggan sangat krusial bagi provider telekomunikasi.*

Pada Tugas Akhir ini akan diterapkan metode hybrid data mining yaitu penggabungan dua metode classification dan clustering. Jaringan Saraf Tiruan (JST) akan digabungkan dengan Self-organizing Map (SOM) untuk menyelesaikan permasalahan prediksi churn pelanggan. SOM akan digunakan sebagai metode clustering untuk mengurangi data yang tidak merepresentasikan data latih, lalu output SOM akan dipakai sebagai data latih prediksi churn pelanggan menggunakan propagasi balik JST.

Penggabungan metode JST dan SOM pada data uji menghasilkan akurasi hingga 96.7%. Berdasarkan hasil uji coba yang dilakukan penggabungan JST-SOM dapat memaksimalkan akurasi klasifikasi.

Kata kunci: Hybrid Data Mining, Self-organizing Map, Jaringan Saraf Tiruan, Churn Pelanggan

IMPLEMENTATION HYBRID NN-SOM METHOD FOR PREDICTION CUSTOMER CHURN: STUDY CASE PT. TELEKOMUNIKASI SELULER

Student's Name	**: Naufal Aulia Rizal**
Student's ID	**: 5110 100 076**
Department	**: Informatics Engineering, FTIF-ITS**
First Advisor	**: Arya Yudhi Wijaya, S.kom., M.kom.**
Second Advisor	**: Rully Soelaiman, S.Kom., M.Kom.**

Abstract

Marketing literature states that it is more costly to engage a new customer than to retain an existing loyal customer. Customer churn is a term used in the telecommunications industry which means customer migration from one provider to another provider. Churn prediction models are developed by academics and practitioners to effectively manage and control customer churn in order to retain existing customer. As churn management is an important activity for companies to retain loyal customer, the abilty to correctly predict customer churn is necessary. As the cellular network services market becoming more competitive, customer churn management has become a crucial task for mobile communication operators

In this final project will apply a hybrid data mining method that combine two method, classification and clustering. Neural Network (NN) combined with Self-organing Map (SOM) will be applied to solve prediction problem for customer churn. SOM will be used for clustering techniques to perform data reduction task by filtering out unrepresentative training data, then the outputs from SOM are used to create the prediction model based on Backpropagation Neural Network.

Hybrid NN and SOM on the test data will generate up to 96.7% accuracy. Based on the test results, the hybrid NN-SOM can maximize classification accuracy.

Keywords: Hybrid Data Mning, Self-organizing Map, Neural Network, Customer Churn.

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "*Implementasi Metode hybrid JST-SOM Pada Prediksi Churn Pelanggan:Studi Kasus PT.Telkomsel*" dengan tepat waktu.

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata bagi kampus Teknik Informatika, ITS, dan bangsa Indonesia.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku dosen pembimbing penulis yang telah memberikan bimbingan, saran, kritik, dan ilmu yang sangat bermanfaat hingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Arya Yudhi Wijaya, S.kom., M.kom**.** selaku dosen pembimbing yang telah memberikan nasihat, arahan, dan bimbingan dengan penuh kesabaran sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Ibu Dr. Nanik Suciati selaku ketua jurusan Teknik Informatika ITS, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya
4. Gregorius Edwadr dan Nabil yang telah membantu memberi masukan dan gagasan dalam proses pengerjaan Tugas Akhir.
5. Keluarga dan teman-teman penulis yang waktunya tersita selama proses pengerjaan Tugas Akhir ini.
6. Juga tak lupa kepada semua pihak yang belum sempat disebutkan satu per satu di sini yang telah membantu terselesaikannya tugas akhir ini.

Tugas Akhir ini merupakan persembahan penulis untuk kedua orang tua penulis yang selalu mengingatkan untuk menuntut ilmu setinggi-tingginya.

Kesempurnaan tentu masih jauh tercapai pada Tugas Akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk perbaikan ke depan. Semoga Tugas Akhir ini dapat bermanfaat bagi perkembangan ilmu pengetahuan dan bagi semua pihak.

Akhir kata, mohon maaf yang sebesar-besarnya jika terdapat kesalahan pada buku Tugas Akhir ini. Selain itu, penulis juga mengharapkan kritik dan saran yang membangun.

Surabaya, Juni 2014

Naufal Aulia Rizal

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Industri seluler di Indonesia berkembang sangat pesat. Perkembangan itu ditandai juga dengan tingginya *churn rate* atau perpindahan pelanggan dari satu *provider* ke *provider* lain. Jadi, bila pada akhir tahun suatu *provider* memiliki 100 juta pelanggan dengan *churn rate* 2%, itu artinya pada tahun tersebut ada 2 juta pelanggan yang berpindah *provider*. *Churn rate* adalah indikator kemapanan suatu *provider* makin rendah *churn rate*, semakin bagus kinerja suatu *provider*. Untuk menurunkan *churn rate* perlu dilakukan tindakan pencegahan salah satunya dengan membuat model prediksi untuk mengetahui pelanggan mana yang berpotensi *churn*. Dengan memprediksi pelanggan yang berpotensi *churn*, *provider* dapat mengurangi *churn rate* dengan menawarkan program-program retensi baru yang membuat pelanggan tetap bertahan. Hal tersebut sangat penting karena biaya untuk mendapat satu pelanggan sangat jauh lebih tinggi daripada mempertahan satu pelanggan.

Permasalahan utama dalam prediksi *churn* adalah kurang akuratnya model prediksi. Prediksi *churn* akan sia-sia jika hasil prediksi yang tidak akurat karena akan mengakibatkan *provider* menghabiskan biaya untuk men-*treatment* pelanggan yang sebenarnya tidak berpotensi *churn*. Untuk itu perlu dibuat model prediksi *customer churn* yang efektif dan akurat dengan menggunakan metode *data mining*. Metode *data mining* dapat digunakan untuk mendeskripsikan pola atau hubungan data dan juga dapat digunakan untuk memprediksi atau menklasifikasi perilaku model berdasarkan data yang tersedia.

Salah satu algoritma yang diaplikasikan adalah Jaringan Saraf Tiruan *(*JST). JST dapat diaplikasikan untuk memprediksi masalah *Customer Churn,* di sisi lain metode *hybrid data mining* yaitu metode untuk menggabungkan 2 atau lebih algoritma yang berbeda dapat meningkatkan performa dari pada metode lain pada

domain yang berbeda [2]. Sehubungan dengan hal tersebut dalam Tugas Akhir ini menerapkan metode *hybrid data mining* yaitu penggabungan metode propagasi balik JST dengan *Self-organizing map* (SOM). Fungsi dari SOM adalah mengurangi data dengan cara menyaring data yang tidak merepresentasikan data latih, lalu *output* dari SOM digunakan untuk membuat model prediksi *churn* memakai JST.

Untuk mengevaluasi metode JST-SOM, dalam Tugas Akhir ini akan membandingkan metode JST-SOM dengan metode klasifikasi JST. Hasil dari eksperimen pada Tugas Akhir ini diharapkan JST-SOM mempunyai akurasi yang lebih tinggi daripada JST.

1.2 Rumusan Masalah

Rumusan masalah yang dapat diangkat dalam Tugas Akhir ini adalah:

1. Proses penggabungan metode JST dan Self-Organizing Map (SOM) sehingga menghasilkan metode hybrid data mining yang baik.
2. Metode JST dan SOM dapat digunakan untuk memprediksi customer churn.
3. Pengurangan data menggunakan SOM untuk meningkatkan akurasi JST.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Metode yang digunakan adalah JST digabung dengan algoritma SOM.
2. Algoritma SOM digunakan untuk pengurangan data.
3. *Map* SOM diset berukuran 2z2.
4. Dataset didapat dari PT. Telekomunikasi Seluler (TELKOMSEL) pada periode November 2013-Januari 2014.

1.4 Tujuan dan Manfaat

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membangun sebuah metode gabungan antara JST-SOM.
2. Membuktikan bahwa implementasi metode JST-SOM dapat diaplikasikan untuk menprediksi customer churn
3. Membuktikan bahwa pengurangan data menggunakan SOM dapat meningkatkan akurasi JST.

Tugas Akhir ini diharapkan dapat meningkatkan akurasi pada prediksi *customer churn* untuk meminimalkan tingkat *churn* pelanggan di PT.Telekomunikasi Seluler. Peningkatan akurasi prediksi *customer churn* dilakukan menggunakan model JST yang dikombinasikan dengan SOM.

1.5 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan untuk mengimplementasikan metode *hybrid* JST-SOM dalam klasifikasi *churn* pelanggan Telkomsel.

2. Studi literatur
   Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang diperlukan untuk mengiimplementasikan metode *hybrid* JST-SOM. Dasar informasi yang diperlukan pada pembuatan implementasi ini di antaranya mengenai metode *Artificial Neural Network* secara umum, metode *self-organizng map* dan

metode *backpropagation Neural Network* secara spesifik. Informasi dan literatur didapatkan dari jurnal-jurnal di internet dan buku.

3. Perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan sistem yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah sistem yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.6 Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. DASAR TEORI

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

BAB III. PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk diagram alir dan *pseudocode*.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

BAB VI. KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan.

2.1 Prediksi Churn

Customer Churn adalah istilah yang digunakan di dalam dunia industri telekomunikasi selular yang artinya adalah perpindahan pelanggan dari provider satu ke provider lainya. Dari segi marketing, sangat penting untuk mengelola *churn* karena biaya untuk mendapatkan pelanggan baru jauh lebih besar daripada mempertahankan pelanggan lama sehingga strategi marketing kedepannya bagi perusahaan telekomunikasi adalah mempertahankan pelanggan dan menghindari *customer churn* [1].

Burez dan Van den Poel (2007) menyatakan tedapat dua tipe pendekatan untuk mengelola *customer churn* yaitu *reactive* dan *proactive*. Ketika perusahaan memakai pendekatan *reactive*, perusahaan akan menunggu pelanggan untuk meminta penghentian pelayanan setelah itu baru perusahaan akan menawarkan suatu program agar pelanggan tersebut tetap bertahan. Ketika perusahaan memakai pendekatan *proactive* perusahaan akan mencoba untuk memprediksi pelanggan mana yang berpotensi *churn* sebelum pelanggan tersebut *churn* sehingga dapat diberi perlakuan khusus dengan suatu program agar pelanggan tersebut tetap bertahan. Tetapi, pendekatan tersebut menjadi tidak efektif apabila *churn prediction* tidak akurat karena perusahaan akan membuang-buang uang kepada pelanggan yang sebenarnya tidak berpotensi *churn*. Sehingga sangat penting untuk membuat model prediksi yang baik untuk menghindari hal tersebut [2].

Churn prediction adalah salah satu *task data mining* yang bertujuan untuk memprediksi pelanggan yang berpotensi *churn* sehingga pelanggan yang berpotensi *churn* dapat segara

diperlakukan secara khusus untuk mencegah terjadinya *churn*. Terdapat dua tujuan dari *data mining* yaitu deskripsi dan prediksi. Deskripsi adalah mencari pola yang mendeskripsikan suatu data, dan prediksi adalah pemakaian variabel-variabel yang diambil dari *dataset* untuk memprediksi nilai yang akan datang. Tujuan tersebut dapat dilakukan menggunakan metode *data mining* seperti *classification*, dan *clustering* [1].

2.2 Klasifikasi

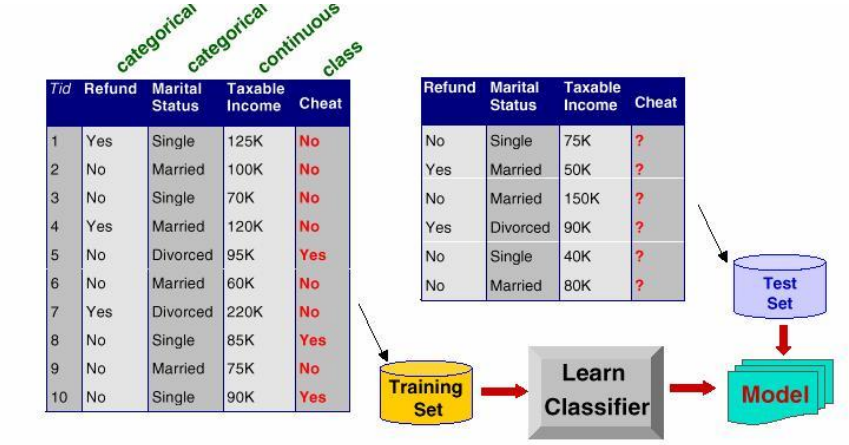
Klasifikasi adalah pengelompokan beberapa objek ke dalam satu atau beberapa kategori yang telah didefinisikan. Proses pengelompokan tersebut didasarkan pada model yang dibentuk dari hasil pembelajaran (*learning*) data yang sudah diketahui kategorinya. Klasifikasi bisa digunakan pada banyak kasus, misalnya memprediksi tumor kanker apakah *benign* atau *malignant*, klasifikasi penggunaan kartu kredit yang sah atau tidak, mengklasifikasi topik berita menjadi bisnis, hiburan, olahraga dan lainnya [2]. Pada Tugas Akhir ini klasifikasi digunakan untuk memprediksi pelanggan apakah *churn* atau tidak.

Ilustrasi proses klasifikasi dapat dilihat pada Gambar 2.1. Data masukan dalam proses klasifikasi adalah sekelompok *record*. Masing-masing *record* memiliki satu *tuple* (x, y) di mana *x* adalah himpunan atribut atau variabel prediktor dan *y* adalah kelas atau kategori data [2].

Pada proses klasifikasi, data masukan dibagi menjadi dua, yakni data latih (*training*) dan data uji (*testing*). Data latih adalah data yang digunakan untuk membentuk model klasifikasi, sedangkan data uji adalah data yang digunakan untuk menguji performa dari model klasifikasi.

Algoritma klasifikasi bisa dibagi menjadi dua jenis berdasarkan algoritma pembelajarannya. Yaitu *lazy learning* dan *eager learning*. Algoritma *lazy learning* berarti menyimpan data latih sebagai model yang nantinya digunakan untuk tahap uji. Sedangkan algoritma *eager learning* akan membuat suatu model

klasifikasi dari data latih yang ada dan nantinya digunakan dalam tahap uji untuk klasifikasi [2].



Gambar 2.1 Ilustrasi Proses Klasifikasi

Algoritma *lazy learning* sendiri masih dibagi dalam beberapa jenis pembelajaran. Beberapa diantaranya adalah pembelajaran berbasis *instance* dan pembelajaran berbasis *case*. Sedangkan *eager learning* bisa dibagi menjadi pembelajaran berbasis model yaitu membuat model klasifikasi seperti jaringan saraf buatan, pembelajaran berbasis statistik yaitu membuat model klasifikasi dengan penghitungan statistik seperti *naive bayes classifier* dan pembelajaran berbasis *rule* yaitu membuat model klasifikasi dengan menggunakan aturan *rule* seperti *reduced error pruning* [2].

2.3 Identifikasi Data Noise

Data *noise* diartikan sebagai data yang tidak berarti dan bisa menyebabkan nilai akurasi menjadi rendah sehingga data semacam ini bisa dihilangkan. Regresi dan *clustering* adalah beberapa

metode yang bisa digunakan untuk mengidentifikasi *noise (*Han, Kamber, & Pei, 2012).

Pada Tugas Akhir ini *Self-organizing Map* yaitu metode *clustering* digunakan untuk menghilangkan *noise* yaitu data yang tidak merepresentasikan data latih. Metode *Self-organizing map* diharapkan dapat meningkatkan akurasi dari model prediksi Jaringan Saraf Tiruan.

2.4 Normalisasi

Satuan yang digunakan oleh atribut bisa mempengaruhi analisis data. Misalnya, mengubah satuan ukuran dari meter menjadi inci untuk tinggi, atau dari kilogram menjadi pon untuk berat, bisa menghasilkan hasil yang berbeda. Untuk membantu menghindari ketergantungan pada pilihan satuan ukuran, maka data dinormalisasi. Yaitu mengubah nilai data dalam suatu rentang seperti (-1, 1) atau (0, 1) [2].

Normalisasi data bertujuan agar semua atribut mempunyai rentang nilai yang sama. Normalisasi mencegah atribut yang memiliki rentang data luas bernilai lebih besar daripada atribut dengan rentang data kecil. Normalisasi khususnya berguna untuk algoritma klasifikasi seperti jaringan saraf tiruan dan klasifikasi berbasis jarak [2].

Ada beberapa metode untuk normalisasi data, yaitu:

1. Normalisasi min-max

$$v_i' = \frac{v_i - min_A}{max_A - min_A} \tag{2.1}$$

dimana v_i adalah nilai data ke-i pada atribut A saat ini, min_A adalah nilai paling kecil pada data atribut A dan max_A adalah nilai paling besar pada data atribut A.

2. Normalisasi *z-score*

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A} \tag{2.2}$$

Dimana v_i adalah nilai data ke-*i* pada atribut *A* saat ini, $\bar{A}$ adalah nilai rata-rata data atribut A dan σ_A adalah standar deviasi data atribut A.

2.5 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) merupakan kategori ilmu *soft computing* yang memiliki karakteristik mirip dengan saraf biologi yang mampu memberikan stimulasi, melakukan proses, dan memberikan *output*. Jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan proses perhitungan selama proses pembelajaran [4].

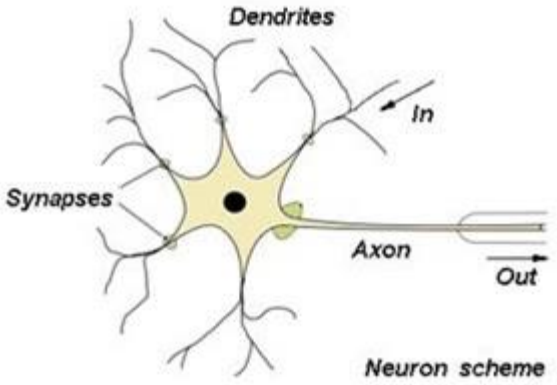
Jaringan Saraf Tiruan terdiri dari elemen sederhana yang dinamakan *neuron*. Setiap *neuron* terkoneksi dengan *neuron* lainnya dengan bobot penghubung. Bobot merepresentasikan informasi yang digunakan oleh jaringan untuk menyelesaikan masalah. Setiap *neuron* mempunyai fungsi aktivasi yang merupakan fungsi dari *input* yang telah diterima. *Neuron* mengirimkan aktivasinya sebagai sebuah sinyal ke beberapa *neuron* yang lain. Ciri utama yang dimiliki JST adalah kemampuan untuk belajar. Belajar pada JST dapat diartikan sebagai proses penyesuaian parameter pembobot karena nilai *output* yang didapat bergantung dengan nilai bobot yang menghubungkan antara *neuron* satu dengan *neuron* lainnya.

Fungsi dari Neural Network diantaranya adalah:

1. Pengklasifikasian pola
2. Memetakan pola yang didapat dari input ke dalam pola baru pada output
3. Penyimpan pola yang akan dipanggil kembali
4. Memetakan pola-pola yang sejenis
5. Pengoptimasi permasalahan
6. Prediksi

2.5.1 Struktur Dasar Jaringan Biologi

Pembuatan struktur jadingan saraf tiruan diilhami oleh struktur jaringan biologi, khususnya jaringan otak manusia. Untuk lebih mengenal asal-usul serta bagaimana suatu struktur jaringan saraf tiruan dibuat dan dapat dipakai sebagai *machine learning* [4]. Berikut ini akan dijelaskan istilah yang secara umum digunakan.

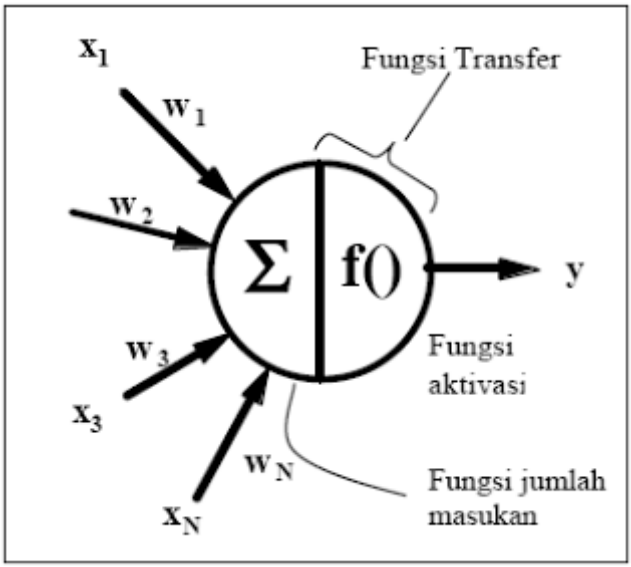


Gambar 2.2 Struktur Dasar Jaringan Saraf

Neuron adalah satuan unit pemproses terkecil pada otak, bentuk sederhana sebuah *neuron* digambarkan pada Gambar 2.2. Struktur pada Gambar 2.2 adalah bentuk standar satuan unit jaringan otak manusia. Jaringan otak manusia tersusun tidak kurang dari 10^{13} *neuron* yang masing masing terhubung oleh sekitar 10^{15} dendrit. Fungsi dendrit adalah sebagai penyampaian sinyal dari *neuron* tersebut ke *neuron* yang terhubung dengannya. Sebagai keluaran, setiap *neuron* memiliki *axon*, sedangkan bagian penerima sinyal disebut sinapsis. Secara umum jaringan saraf terbentuk dari jutaan struktur dasar *neuron* yang terinterkoneksi dan terintegrasi antara satu dengan yang lain sehingga dapat melaksanakan aktifitas secara teratur dan terus menerus sesuai kebutuhan.

2.5.2 Konsep Dasar Pemodelan JST

Tiruan *neuron* dalam struktur jaringan saraf tiruan adalah sebagai elemen pemproses seperti pada gambar 2.3 yang dapat berfungsi seperti halnya sebuah *neuron*.



Gambar 2.3 Model *Neuron*

Sejumlah sinyal masukan *a* dikalikan dengan masing-masing yang bersesuaian *w*. Kemudian dilakukan penjumlahan dari seluruh hasil perkalian tersebut dan keluaran yang dihasilkan dilakukan kedalam fungsi aktivasi untuk mendapatkan tingkatan derajat sinyal keluaran.

Kumpulan dari *neuron-neuron* dibuat menjadi sebuah jaringan yang akan berfungsi sebagai alat komputasi. Jumlah *neuron* dan struktur jaringan untuk setiap problem yang akan dihasilkan adalah berbeda.

2.5.3 Proses Pembelajaran

2.5.3.1 Pembelajaran Terawasi

Metode pembelajaran pada jaringan syaraf disebut terawasi jika nilai keluaran yang diharapkan telah diketahui sebelumnya. Salah satu model JST yang menggunakan metode pembelajaran terawasi adalah *backpropagation neural network* (BPNN).

Algoritma pembelajaran BPNN melakukan perubahan bobot-bobot yang menghubungkan antar *neuron* dengan menggunakan nilai *error* yang didapat dari perhitungan maju [4].

2.5.3.2 Pembelajaran Tidak Terawasi

Pada metode pembelajaran tak terawasi ini tidak memerlukan target *output*. Tujuan metode ini adalah pengelompokan unit-unit yang hampir sama dalam suatu area tertentu. Salah satu metode JST yang menggunakan metode pembelajaran tak terawasi adalah *Self-organizing Map* (SOM).

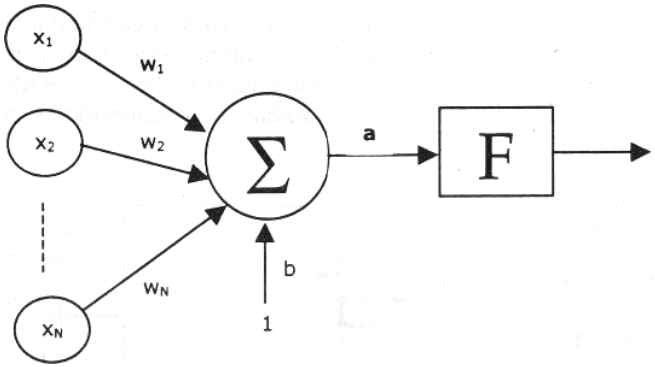
Pada jaringan SOM, suatu lapisan yang berisi *neuron* akan menyusun dirinya sendiri berdasarkan input nilai tertentu dalam suatu kelompok yang dikenal dengan istilah *cluster*.

2.5.4 Fungsi Aktivasi

Jaringan syaraf merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka *neuron* tersebut akan diaktifkan, tapi kalau tidak, maka *neuron* tersebut tidak akan diaktifkan. Apabila *neuron* tersebut diaktifkan, maka *neuron* tersebut akan mengirimkan *output* melalui bobot-bobot *output* nya ke semua *neuron* yang berhubungan dengannya. Demikian seterusnya.

Pada jaringan syaraf, *neuron-neuron* akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan *neuron* (*neuron layers*). Biasanya *neuron-neuron* pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan lapisan *output*). Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya, yang sering dikenal dengan nama lapisan tersembunyi (*hidden layer*). Gambar 2.4 menunjukkan *neuron* dengan masukan data $(x_1, x_2, \dots, x_n)$ dengan masing-masing bobot $(w_1, w_2, \dots, w_n)$ dan bobot bias *b*. dan fungsi aktivasi F yang akan menjadi keluaran dari *neuron*.



Gambar 2.4 Fungsi Aktivasi Jaringan Sederhana

2.6 Arsitektur Jaringan Saraf Tiruan

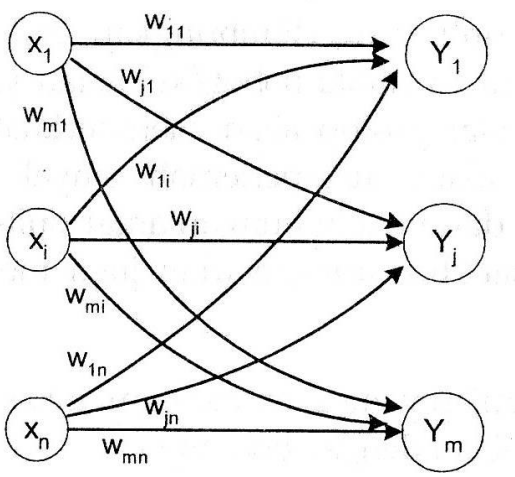
Berdasarkan arsitekturnya jaringan saraf tiruan digolongkan menjadi:

2.6.1 Jaringan Layar Tunggal

JST dengan layar tunggal pertamakali dirancang oleh Widrow dan Hoff pada tahun 1960. Walaupun JST layar tunggal ini sangat terbatas penggunaannya, namun konsep dan gagasannya

banyak dipakai oleh beberapa pakar untuk membuat model JST layar jamak.

Dalam jaringan ini, sekumpulan *input neuron* dihubungkan langsung dengan sekumpulan *output*. Dalam beberapa model (misal perceptron), hanya ada sebuah unit *neuron output*.



Gambar 2.5 Jaringan Layar Tunggal

Pada Gambar 2.5 menunjukkan arsitektur jaringan saraf tiruan dengan layar tunggal.

Semua unit *input* dihubungkan dengan semua unit *output*, meskipun dengan bobot yang berbeda-beda. Tidak ada unit *input* yang dihubungkan dengan unit *input* lainnya. Demikian pula dengan unit *output*.

Besarnya w_{ji} menyatakan bobot hubungan antara unit ke-*i* dalam *input* dengan unit ke-*j* dalam *output*. Bobot-bobot ini saling independen. Selama proses pelatihan, bobot-bobot tersebut akan dimodifikasi untuk meningkatkan keakuratan hasil.

2.6.1.1 Input Layer

Input Layer atau bisa juga disebut sebagai lapisan masukan adalah lapisan yang dimana mempunyai jumlah *neuron* yang sama dengan jumlah *input*. Data yang menjadi masukan pada lapisan ini juga menjadi keluaran dari lapisan ini juga atau dengan kata lain tidak ada fungsi aktifasi pada lapisan ini.

2.6.1.2 Output Layer

Output Layer atau lapisan keluaran adalah lapisan yang berada di paling akhir. Jumlah *neuron* pada lapisan ini akan berjumlah satu jika hanya ada dua kelas pada data asli, atau sama dengan jumlah kelas pada asli jika terdapat lebih dari dua kelas pada data asli

2.6.2 Jaringan Layar Jamak

Multi Layer Perceptron adalah salah satu tipe dari *Neural Network* yang terdiri dari sejumlah neuron yang dihubungkan oleh bobot-bobot penghubung. *Neuron-neuron* tersebut disusun dalam lapisan-lapisan yang terdiri dari satu lapisan masukan (*input layer*), satu atau lebih lapisan tersembunyi (*hidden layer*), dan satu lapisan keluaran (*output layer*). Gambar 2.6 menunjukkan arsitektur jaringan saraf tiruan dengan layar jamak.

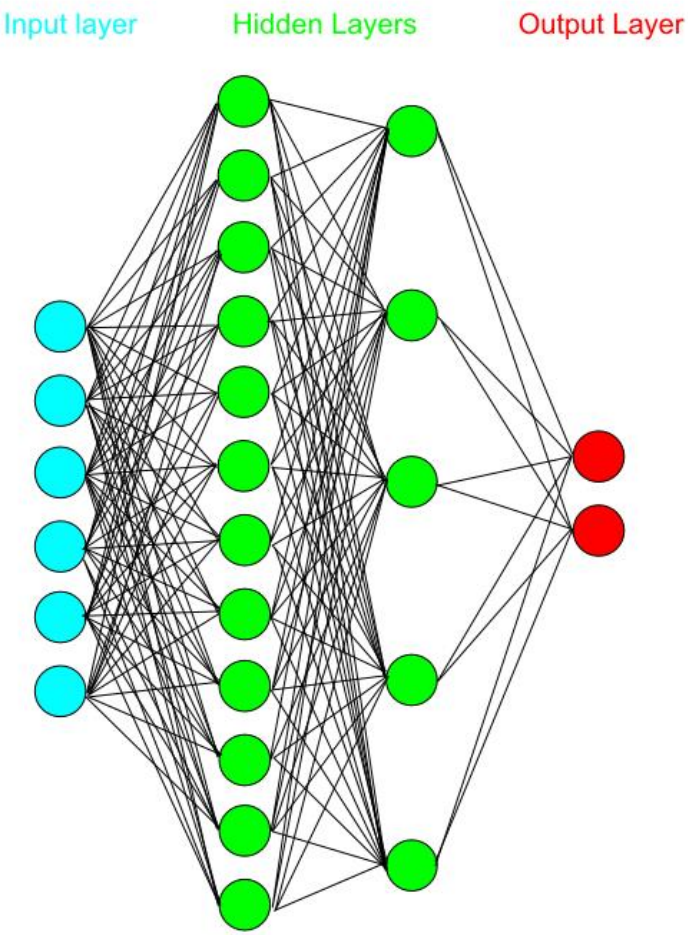
2.6.2.1 Input Layer

Input Layer atau bisa juga disebut sebagai lapisan masukan adalah lapisan yang dimana mempunyai jumlah *neuron* yang sama dengan jumlah *input*. Data yang menjadi masukan pada lapisan ini juga menjadi keluaran dari lapisan ini juga atau dengan kata lain tidak ada fungsi aktivasi pada lapisan ini.

2.6.2.2 Hidden Layer

Hidden Layer atau lapisan tersembunyi adalah lapisan yang berada di antara lapisan masukan dan lapisan keluaran. Tidak ada batasan banyaknya lapisan tersembunyi dan jumlah *neuron* pada setiap lapisan. Setiap *neuron* pada lapisan masukan terhubung dengan setiap *neuron* pada lapisan tersembunyi.

Demikian juga, setiap *neuron* pada lapisan tersembunyi terhubung ke setiap *neuron* pada lapisan keluaran.



Gambar 2.6 Arsitektur Multi Layer Perceptron

2.6.2.3 Output Layer

Output Layer atau lapisan keluaran adalah lapisan yang berada di paling akhir. Jumlah *neuron* pada lapisan ini akan berjumlah satu jika hanya ada dua kelas pada data asli, atau sama dengan jumlah kelas pada asli jika terdapat lebih dari dua kelas pada data asli

2.7 Backpropagation Neural Network

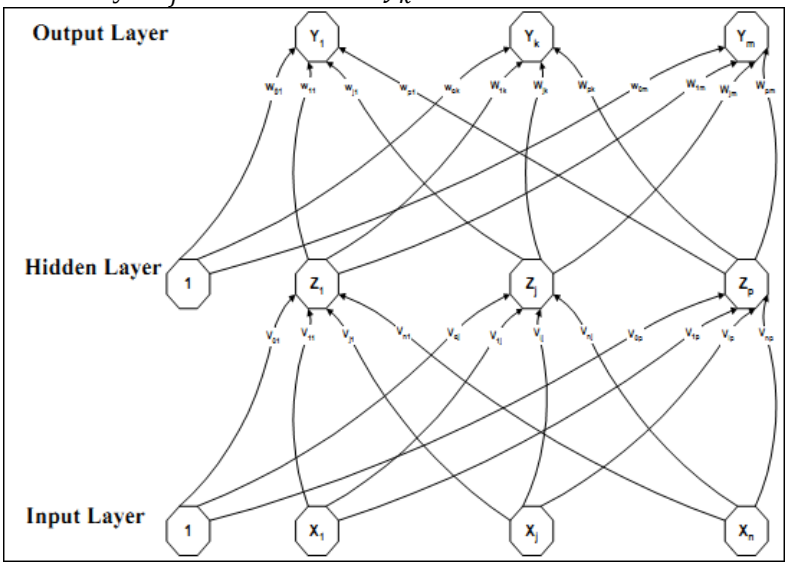
Back Propagation Neural Network (BPNN) adalah suatu metode Jaringan Saraf Tiruan (JST) yang menggunakan pelatihan *supervised* dan didesain untuk operasi pada jaringan *multi layer perceptron* untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang terdapat pada lapisan tersembunyi. Algoritma BPNN menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur *(backward*). Sebelum mendapatkan *error output* terlebih dahulu dilakukan perambatan maju (*feed forward*) [4].

Pemilihan bobot awal sangat mempengaruhi jaringan syaraf tiruan dalam mencapai *minimum global* (atau mungkin lokal saja) terhadap nilai error (kesalahan) dan cepat tidaknya proses pelatihan menuju kekonvergenan. Apabila bobot awal terlalu besar maka *input* (masukan) ke setiap lapisan tersembunyi atau lapisan output (keluaran) akan jatuh pada daerah dimana turunan fungsi sigmoidnya akan sangat kecil. Apabila bobot awal terlalu kecil, maka *input* (masukan) ke setiap lapisan tersembunyi atau lapisan *output* (keluaran) akan sangat kecil. Hal ini akan menyebabkan proses pelatihan berjalan sangat lambat. Biasanya bobot awal diinisialisasi secara *random* dengan nilai antara -0.5 sampai 0.5 atau (-1 sampai 1).

2.7.1 Arsitektur BPNN

Gambar 2.7 adalah arsitektur BPNN dengan *n* buah *neuron* masukan ditambah dengan bias, sebuah layar tersembunyi yang terdiri dari *p* unit hidden layer ditambah dengan bias, serta *m* buah unit keluaran. v_{ij} merupakan bobot yang menghubungkan unit

masukan x_i ke unit hidden layer z_j. w_{kj} merupakan bobot dari unit hidden layer z_j ke unit keluaran y_k.



Gambar 2.7 Arsitektur Jaringan BPNN

2.7.2 Skema Pembelejaran

Terdapat 3 fase pelatihan pada metode BPNN [5], yaitu:

1. Propagasi Maju

   Selama propagasi maju sinyal masukan x_i dipropagasikan ke layar tersembunyi lalu diaktivasi menggunakan fungsi aktivasi. Keluaran dari setiap unit pada layar tersembnyi selanjutnya dipropagasikan lagi ke layar keluaran y_k.

   Setelah itu, keluaran jaringan y_k dibandingkan dengan target dari dari data sebenarnya. Selisih dari hasil propagasi maju dengan data sebenarnya adalah *error* yang didapat. *Error* tersebut nantinya akan digunakan untuk perubahan bobot yang dijelaskan pada fase perubahan bobot.

2. Propagasi Mundur
   Berdasarkan *error* pada layar *output*, dihitung faktor yang dipakai untuk mendistribusikan *error* di unit *output* ke semua unit di layar tersembunyi yang terhubung dengan unit *output* di layar keluaran.

3. Perubahan Bobot
   Setelah semua faktor *error* dihitung, bobot yang menghubungkan tiap-tiap unit pada *hidden layer* dan *output layer* diubah secara bersamaan.

2.7.3 Fungsi Aktivasi

Fungsi Aktivasi dalam BPNN harus memenuhi beberapa syarat yaitu: kontinu, terdiferensial dengan mudah, dan merupakan fungsi yang tidak turun. Salah satu fungsi aktivasi yang memenuhi syarat tersebut dan sering diimplementasikan pada BPNN adalah sigmoid biner yang memiliki rentang (0, 1) dan sigmoid bipolar yang memiliki rentang (-1, 1).

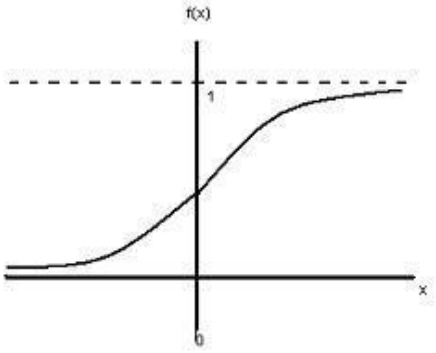
Pada Gambar 2.8, bentuk sigmoid biner yang ada pada Persamaan 2.3 dan Persamaan 2.4

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$f'(x) = f(x)(1 - f(x)) \tag{2.4}$$

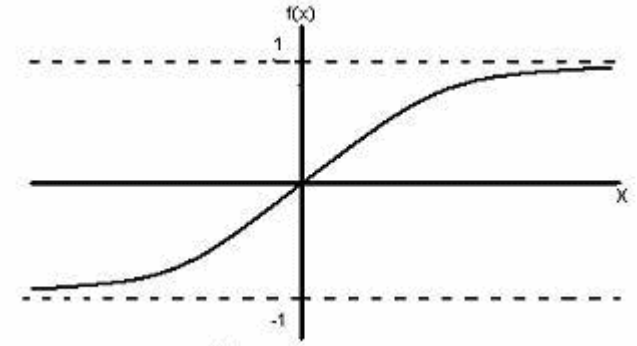
Pada Gambar 2.9, bentuk sigmoid bipolar yang ada pada Persamaan 2.5 dan Persamaan 2.6.

$$f(x) = \frac{2}{1 + \exp(-z)} - 1 \tag{2.5}$$



Gambar 2.8 Fungsi Sigmoid Biner

$$f'(x) = \frac{1}{2}(1 + f(x))(1 - f(x)) \quad (2.6)$$



Gambar 2.9 Funsi Aktivas Sigmoid Bipolar

2.7.4 Algoritma Pembelajaran

Algoritma pembelajaran BPNN dengan satu hidden layer adalah sebagai berikut:

1. Inisialisasi bobot awal dengan nilai random kecil antara (-1.1).

2. Jika kondisi penghentian belum dipenuhi lakukan langkah 2-8.
3. Untuk setiap data latih lakukan langkah 3-8.

Fase I: Propagasi maju

4. Tiap unit masukan menerima sinyal dan diteruskan ke unit di *hidden layer*.
5. Hitung semua keluaran unit di *hidden layer*.

$$z_{net_j} = V_{jo} + \sum_{i=1}^{n} x_i w_{ij} \quad (2.7)$$

$$z_j = f\left(z_{net_j}\right) = \frac{1}{1 + e^{-znet_j}} \quad (2.8)$$

6. Hitung semua keluaran unit di *output layer*.

$$y_{net_k} = w_{ko} + \sum_{j=1}^{p} z_j w_{jk} \quad (2.9)$$

$$z_j = f\left(y_{net_k}\right) = \frac{1}{1 + e^{-ynet_k}} \quad (2.10)$$

Fase II: Propagasi mundur

7. Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k (k = 1,2,..., m).

$$\delta_k = (t_k - y_k) f'\left(y_{net_k}\right) t_k = target \quad (2.11)$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di bawahnya (langkah 8). Hitung suku perubahan bobot w_{kj} dengan dengan laju percepatan α.

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.12)$$

8. Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi zj (j = 1, 2, ..., p).

$$\delta_{net_j} = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.13)$$

Faktor δ unit tersembunyi:

$$\delta_j = \delta_{net_j} f'(znet_j) \quad (2.14)$$

Fase II: Perubahan bobot

9. Perubahan bobot yang menghubungkan *hidden layer* dan *output layer*.

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.15)$$

10. Merubah bobot yang menghubungkan *input layer* dan *output layer*.

$$v_{ij}(baru) = v_{ij}(lama) + \Delta w_{ij} \quad (2.16)$$

Setelah proses pembelajaran selesai dilakukan, jaringan dapat dipakai untuk prediksi menggunakan data uji. Dalam hal ini, hanya propagasi maju (langkah 5 dan 6) saja yang dipakai untuk menentukan keluaran jaringan.

Apabila fungsi aktivasi yang dipakai bukan sigmoid biner, maka langkah 5 dan 6 harus disesuaikan. Demikian juga turunannya pada langkah 7 dan 8.

2.8 Self-Organizing Map

Self-organizing map (SOM) merupakan salah satu model jaringan saraf tiruan (JST) yang menggunakan metode *unsupervised learning*. SOM adalah model JST yang dapat digunakan untuk pengelompokan data. Tujuan dari *clustering* adalah mengurangi jumlah data dengan cara mengkategorikan atau mengrupkan data yang sama [6].

SOM terdiri tadi 2 lapisan, yaitu lapisan *input*, dan lapisan *output*. Setiap *neuron* di lapisan *input* terhubung dengan setiap *neuron* pada lapisan *output*. Setiap *neuron* pada lapisan *output* merepresentasikan *cluster* dari *input* yang diberikan.

Metode pelatihan SOM adalah *competition learning*. Ketika data masukan dikenali oleh jaringan, *neuron* di *competition layer* juga ditentukan mana yang terdekat dengan pola masukan. *Neuron* yang terdekat dengan pola masukan disebut *neuron* pemenang. Pada SOM nilai bobot yang diubah bukan hanya bobot yang terhubung dengan neuron pemenang tetapi terdapat hubungan tetangga di layar kompetisi yang mengindikasi bobot *neuron* mana saja yang harus diganti [4].

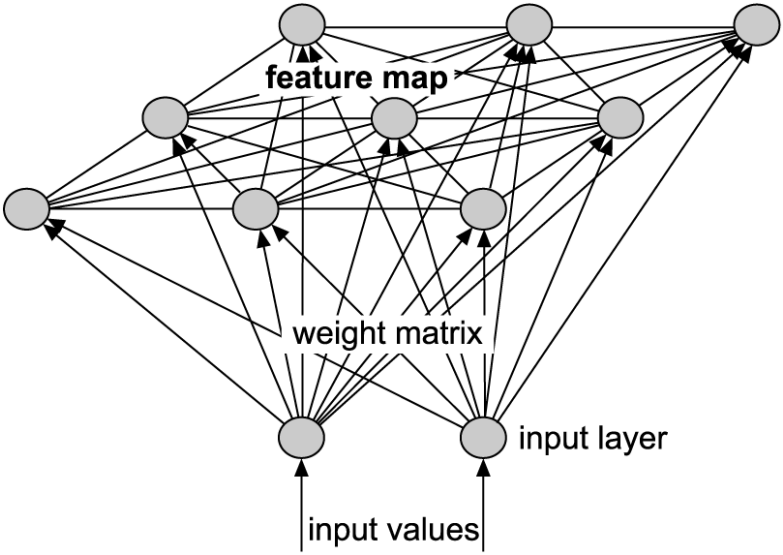
Relasi tetangga pada SOM biasanya direpresentasikan dengan grid 2 dimensi dengan *neuron input* sebagai *vertexnya*. Topologi relasi tetangga biasanya berbentuk *rectangular* atau *hexagonal*.

2.8.1 Aristektur SOM

Gambar 2.10 adalah gambar aristektur SOM. Struktur jaringan SOM merupakan suatu jaringan yang terdiri dari dua lapisan (*layer*), yaitu lapisan *input* dan lapisan *output*, dimana lapisan *input* menyatakan observasi dari data yang terdiri dari *n* elemen (berdimensi-n), sedangkan lapisan *output* merepresentasikan kelompok dari *input* yang diberikan.

Setiap elemen *input* mempunyai bobot untuk masing-masing *output*. Dalam hal ini w_{jk} menyatakan elemen bobot yang menghubungkan elemen input *j* ke *output* k. Setelah *input* dan

output memiliki pasangan terhadap masing-masing bobot, dilakukan pembelajaran untuk mencari nilai bobot yang sesuai.



Gambar 2.10 Arsitektur Self-organizing Maps

2.8.2 Skema Pelatihan

Terdapat 2 fase untuk melakukan *clustering* menggunakan SOM yaitu fase pelatihan dan fase *clustering*. Pada tahap awal pelatihan dilakukan inisialisasi yaitu menetapkan besar *map* SOM untuk menentukan jumlah *cluster* yang diinginkan dan parameter-parameter awal yang akan digunakan, yaitu *learning rate* dan *radius neighborhood*. Data masukan adalah atribut dari dataset yang diset sebagai *input* vektor $x_1 = (x_{11}, x_{12}, \dots, x_{1n})$ dan juga terdapat vektor bobot $w_1 = (w_{11}, w_{12,\dots,} w_{n1})$. Dimana masing masing elemen w_{jk} bernilai diantara rentang 0 dan 1.

Penyesuaian dilakukan dengan melihat kemiripan suatu vektor masukan dengan vektor bobot. Ukuran yang digunakan untuk menyatakan kemiripan antara vektor masukan dan vektor bobot adalah jarak *Euclidian*. Setiap vektor masukan dipilih secara

acak, untuk dihitung jarak *Euclidian*-nya terhadap masing-masing bobot yang telah ditetapkan. Persamaan *Euclidian distance* dijelaskan oleh persamaan 2.17.

$$d = \sqrt{\sum_{j=1}^{n}\left(w_{jk} - x_{ij}\right)^2} \quad (2.17)$$

Dimana w_{jk} adalah nilai vektor bobot dan x_{ij} nilai vektor masukan.

Selanjutnya nilai vektor bobot yang menjadi pemenang diubah. BMU dan vektor-vektor bobot lain yang saling bertetangga dengannya dipindahkan lebih dekat ke vektor masukan. Dengan menggunakan fungsi *neighborhood* ditentukan seberapa jauh perubahan vektor-vektor bobot yang bertetangga. Setelah BMU diubah, proses ini kembali dilakukan terhadap observasi-observasi berikutnya, dan berjalan hingga jumlah iterasi yang ditentukan. Gambar 2.11 menunjukkan BMU dan *neuron-neuron* tetangganya, dan Gambar 2.12 menunjukkan perubahan radius *neighborhood* tiap iterasi pelatihan.

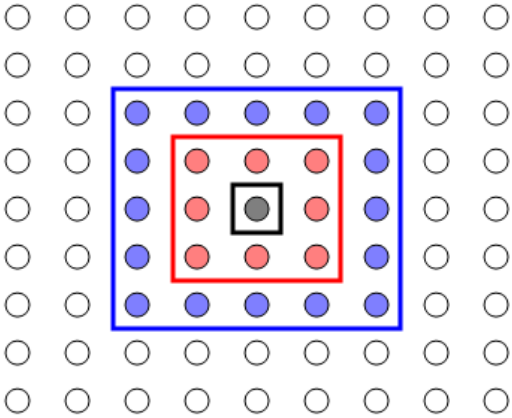
Fase *clustering* dilakukan ketika fase pelatihan telah selesai dengan mendapatkan bobot-bobot *node* yang baru. Pada fase ini setelah menerapkan vektor *input* hanya *neuron* pemenang yang dicari.

2.8.3 Algoritma Pelatihan

1. Inisialisasi bobot dengan nilai random kecil $w_j = [w_{1j}, w_{2j}, \dots w_{nj}]$, set parameter pembelajaran awal, set radius tetangga awal.
2. Tentukan parameter *neighborhood.*
3. Tentukan parameter *learning rate.*
4. Selama kondisi pmberhentian belum terpenuhi lakukan langkah 5 sampai 8.
5. Untuk setiap *input* vektor x, lakukan langkah 6 sampai 8.
6. Untuk setiap j, hitung *distance* dengan perhitungan:

$$D(j) = \sum_i \left(w_{ij} - x_i\right)^2 \quad (2.18)$$

7. Cari indeks J dimana *D(j)* bernilai minimum.
8. Untuk setiap *neighborhood* dari J dan untuk setiap I, lakukan perhitungan:
   $$w_{ij}(new) = w_{ij}(old) + \alpha\left[x_i - w_{ij}(old)\right] \quad (2.19)$$
9. Ubah nilai *learning rate*.
10. Ubah nilai *radius neighborhood*.
11. Cek apakah kondisi pemberhentian sudah dipenuhi, bila *true* lanjut ke langkah 12.
12. Simpan bobot terakhir.



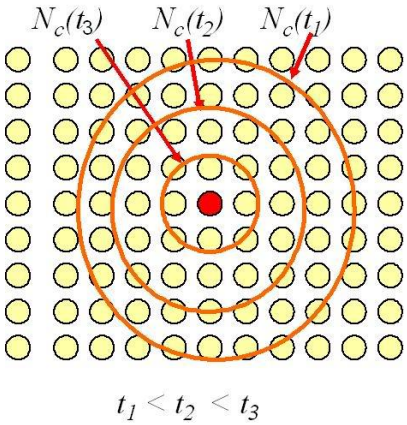
Gambar 2.11 Neuron Pemenang dan neuron tetangganya

Perubahan bobot tidak hanya dikenakan pada *best matching unit* (BMU) tetapi juga ke *neuron* tetangganya, sehingga langkah 8 perlu diubah menggunakan persamaan:

$$m_i(t+1) = m_i(t) + \alpha(t) h_{bi}(t)[x - m_i(t)] \quad (2.20)$$

$$h_{bi}(t) = exp\left(-\frac{\|r_b - r_i\|^2}{2\sigma^2(t)}\right) \quad (2.21)$$

Dimana r_b adalah posisi *neuron* BMU dan r_i adalah posisi pada SOM yang dibuat, dan σ adalah *radius neighborhood.*



Gambar 2.12 Perubahan Radius *Neighborhood*

Untuk menghitung perubahan *learning rate* digunakan persamaan 2.22 sedangkan untuk mengubah *radius neighborhood* digunakan persamaan 2.23.

$$\alpha(t) = \alpha_0 exp\left(-\frac{t}{\lambda}\right) \quad (2.22)$$

$$\sigma = \sigma_0 exp\left(-\frac{t}{\lambda}\right) \quad (2.23)$$

$$\lambda = \frac{T}{\log(\sigma_0)} \quad (2.24)$$

Dimana t adalah iterasi sekarang dan T adalah jumlah maksimal iterasi yang telah di set sebelumnya.

2.8.4 Algoritma Pengelompokan

Proses pengelompokan dengan menggunakan algoritma SOM dilakukan dengan menggunakan algoritma pengenalan SOM. Sama seperti algoritma *neural network* lainnya, algoritma pengenalan SOM merupakan bagian dari algoritma pembelajarannya.

Langkah-langkah fase pengelompkan adalah sebagai berikut:

1. Set nilai bobot w_{ij} (hasil dari proses pelatihan).
2. Untuk setiap *j*, hitung *distance* dengan perhitungan:

$$D(j) = \sum_i \left(w_{ij} - x_i\right)^2 \quad (2.25)$$

3. Cari indeks *J* dimana *D(j)* bernilai minimum.

BAB III
METODOLOGI PERANCANGAN

Pada bab ini akan dijelaskan perancangan sistem prediksi *churn* pelanggan menggunakan metode *hybrid* JST-SOM. Terdapat 2 proses utama yaitu melakukan reduksi pada jumlah data menggunakan metode SOM. Setelah proses implementasi SOM yang telah dilakukan selanjutnya dilakukan prediksi pelanggan *churn/non-churn* menggunakan metode JST. Sebelumnya pada bab ini akan dijelaskan gambaran umum program utama dalam bentuk diagram alir.

3.1 Lingkungan Perancangan Perangkat Lunak

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam perancangan aplikasi adalah:

- Perangkat keras

Uji coba dilakukan pada sebuah PC dengan spesifikasi Processor AMD A6-3650 (2.90 GHz, 8.00 GB RAM), 64-bit Operating System.

- Perangkat lunak

Perangkat lunak ini dikembangkan pada sistem operasi Windows 8 Ultimate dengan menggunakan Matlab 2011a.

3.2 Perancangan Data

Perancangan data merupakan hal penting untuk diperhatikan karena diperlukan data yang tepat agar sistem berjalan dengan baik. Data yang dibutuhkan untuk melakukan klasifikasi ada dua yaitu, data masukan dan data keluaran.

3.2.1 Data Masukan

Data masukan untuk implementasi JST-SOM adalah *dataset* pelanggan dengan jumlah record 1000 dan jumlah fitur 64 atribut.

Pada uji coba masing-masing *dataset* akan dibagi menjadi data latih dan data uji. Pembagian data latih dan data uji dilakukan dengan memberikan besarnya proporsi pada *dataset* seperti 60% data latih dan 40% data uji.

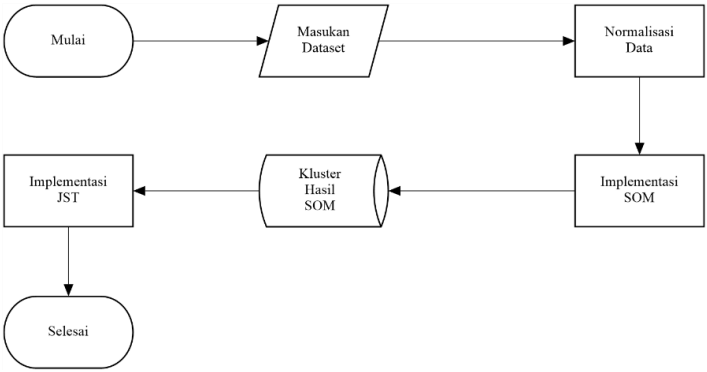
3.2.2 Data Keluaran

Data keluaran merupakan hasil dari proses. Data keluaran dari SOM adalah *dataset* pelanggan yang telah dikelompokkan. Data keluaran dari proses latih JST adalah berupa model prediksi yang bisa dipakai untuk menguji data uji pada proses uji. Adapun data keluaran dari proses uji adalah hasil prediksi *churn* terhadap *dataset* Pelangan.

3.3 Perancangan Proses

3.3.1 Perancangan Proses Secara Umum

Metode hybrid JST-SOM menerapkan penggabungan metode klasifikasi dan *clustering*. Metode SOM digunakan sebagai teknik *clustering* untuk melakukan pengurangan data pada *dataset* yang nantinya 2 kluster yang memiliki proposi *churn* dan *non-churn* terbanyak akan digunakan sebagai data latih BPNN.



Gambar 3.1 Diagram Alir JST-SOM

3.3.2 Normalisasi Data

Normalisasi data bertujuan agar semua atribut mempunyai rentang nilai yang sama. Normalisasi mencegah atribut yang memiliki rentang data luas bernilai lebih besar daripada atribut dengan rentang data kecil. Normalisasi khususnya berguna untuk algoritma klasifikasi seperti jaringan saraf tiruan. Proses normalisasi data dijelaskan oleh Persamaan 3.1.

$$v_i' = \frac{v_i - min_A}{max_A - min_A} \tag{3.1}$$

dimana v_i adalah nilai data ke-*i* pada atribut *A* saat ini, min_A adalah nilai paling kecil pada data atribut *A* dan max_A adalah nilai paling besar pada data atribut *A*. Kode Normalisasi Data diimplementasikan dengan menggunakan variabel-variabel yang ada pada Tabel 3.1

Tabel 3.1 Variabel yang Digunakan Pada Normalisasi Data

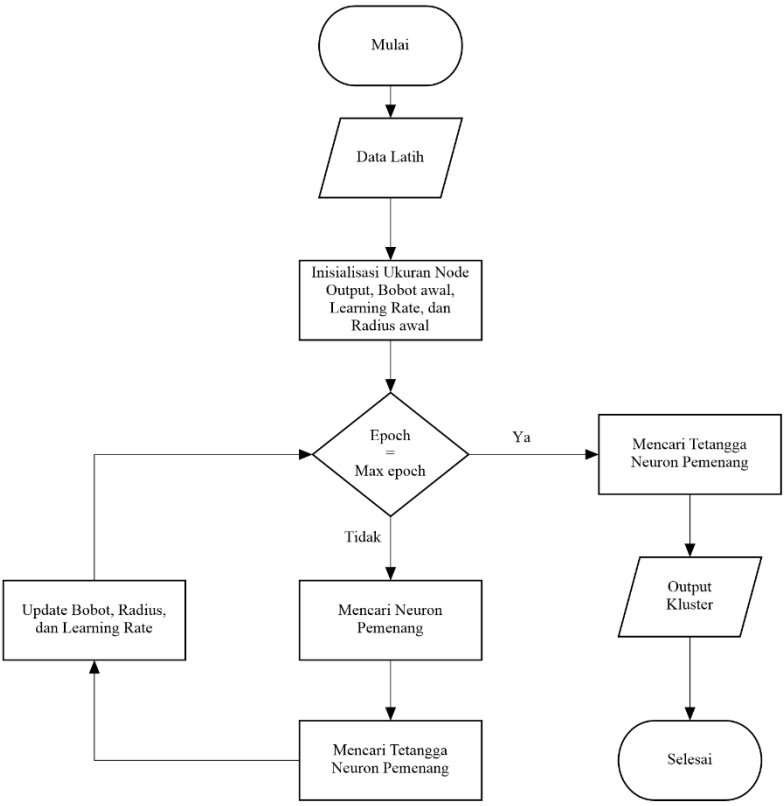
No	Variabel	Tipe	Penjelasan
1	Input	double	atribut data masukan
2	Target	int	Kelas Target
4	num_inputs	int	Jumlah atribut
6	num_targets	int	Jumlah Target
7	num_cases	int	Banyak data

Masukan dari tahap normalisasi data adalah masukan dari *dataset*, dan keluarannya adalah *dataset* yang telah dinormalisasi rentangnya.

3.3.3 Implementasi SOM

Pada bagian ini dijelaskan implementasi SOM untuk melakukan pengurangan jumlah data dengan dilakukan pengelompokan terhadap data masukan. Hasil dari SOM yang digunakan sebagai data latih bagi JST adalah 2 *cluster* yang memiliki jumlah proporsi pelanggan *churn* dan *non-churn*

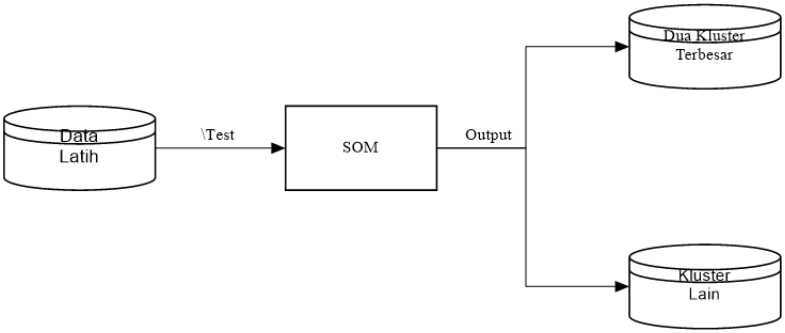
terbanyak. Diagram alir proses implementasi SOM dapat dilihat pada Gambar 3.2



Gambar 3.2 Diagram Alir Implementasi Self-organizing Map

Output dari proses ini adalah *dataset* yang telah dikelompokkan. Jumlah *cluster* pada implementasi SOM adalah sejumlah *map* yang telah ditentukan. Pada gambar 3.3 dijelaskan proses dari implementasi SOM secara detail. Sesuai dengan alur yang telah diilustrasikan pada Gambar 3.2, maka kode SOM dapat

diimplementasikan menggunakan variabel-variabel yang ada pada Tabel 3.2.



Gambar 3.3 Proses Implementasi SOM

Tabel 3.2 Variabel yang digunakan pada Implementasi SOM

No	Variabel	Tipe	Penjelasan
1	trainingSteps	double	Jumlah Iterasi Pelatihan
2	startLearningRate	double	Learning Rate Awal
3	startRadius	double	Radius Tetangga Awal
4	som	double	Nila Bobot
5	totalNeurons	double	Jumlah Neuron Ouput
6	BMU	double	Neuron Pemenang
7	distanceInfluence	double	Radius Neuron Pemenang
8	currentLearning	double	Nilai learning tiap iterasi
9	currentSigma	double	Radius Tiap Iterasi

Langkah-langkah implementasi SOM secara lebih rinci adalah sebagai berikut:

1. Data masukan SOM adalah data latih dari *dataset* pelanggan.

2. Melakukan insialisasi awal untuk ukuran *neuron* keluaran, nilai bobot penghubung *neuron* masukan dengan *neuron* keluaran, nilai *learning rate*, dan nilai besar *radius* tetangga.
3. Jika jumlah iterasi belum maksimal lakukan langkah 4, 5, 6.
4. Mencari *neuron* pemenang yang disebut *best matching unit* (BMU) dilakukan dengan cara mencari jarak paling minimum antara vektor masukan dengan vektor bobot menggunakan *euclidan distance*.
5. Mencari vektor-vektor yang bertetangga dengan BMU.
6. Mungubah nilai bobot BMU dan juga bobot vektor-vektor tetangganya, mengubah *learning rate*, dan menurunkan *radius* tetangga.
7. Setelah iterasi dipenuhi dilakukan pencarian BMU untuk menentukan *cluster*.

3.3.4 Implementasi Jaringan Saraf Tiruan

Pada bagian ini dijelaskan proses implementasi Jaringan Saraf Tiruan (JST). JST pada sistem ini digunakan untuk memprediksi pelanggan mana yang berpotensi *churn* atau *non-churn*. Pelatihan JST menggunakan *back propagation neural network*. Fungsi aktivasi yang digunakan pada implementasi JST adalah fungsi sigmoid biner. Terdapat dua proses pada implementasi JST yaitu proses pelatihan dan proses pengujian.

3.3.4.1 Fungsi Pelatihan

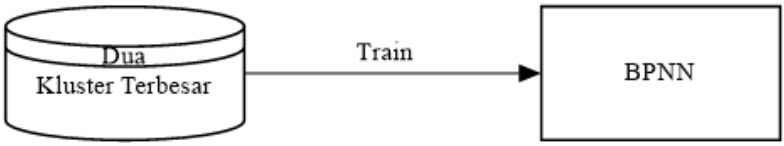
Pada proses pelatihan data masukan adalah hasil dari proses implementasi SOM yaitu 2 *cluster* yang memiliki proporsi *churn* dan *non-churn* terbanyak. Hasil keluaran dari proses pelatihan adalah bobot dan bias yang akan digunakan di proses pengujian. Diagram alir proses pelatihan JST dijelaskan pada Gambar 3.5.

Langkah-langkah implementasi pelatihan BPNN pada Gambar 3.5 secara lebih rinci adalah sebagai berikut:

1. Data masukan untuk data latih pelatihan BPNN adalah data latih yang telah dikelompokkan menggunakan SOM.

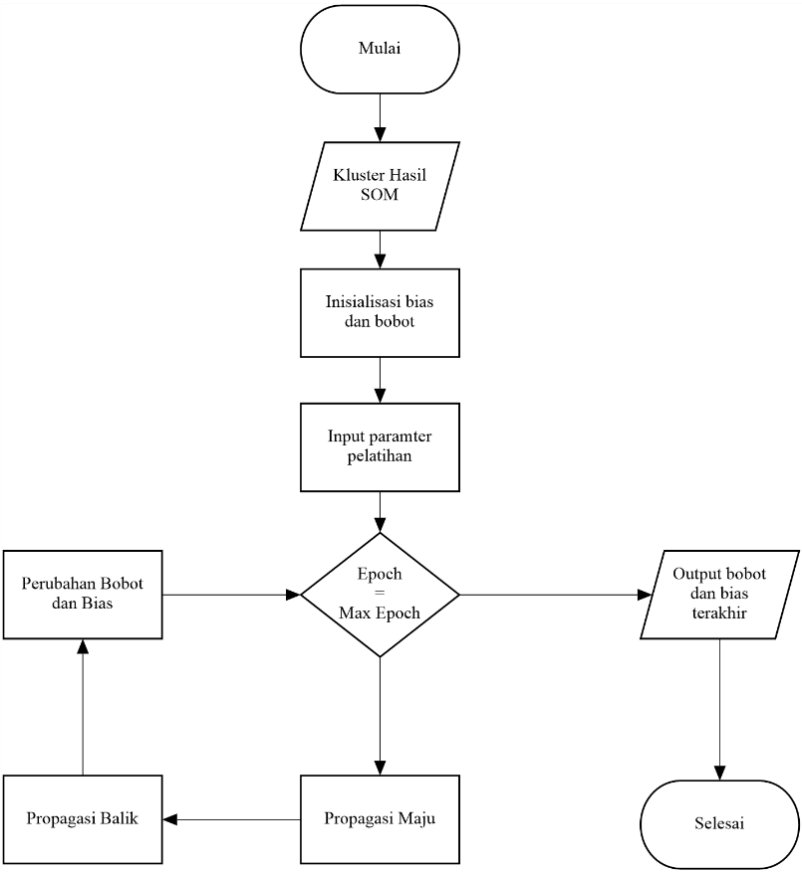
2. Inisialisasi nilai bobot awal adalah nilai *random* kecil diantara rentang (-1, 1) dan nilai bias adalah nilai *random* kecil diantara rentang (-1, 1).
3. Inisialisasi parameter *learning rate* yang nilainya diantara rentang 0 hingga 1, dan inisialiasi jumlah maksimal iterasi pembelajaran.
4. Jika jumlah iterasi pembelajaran belum maksimal lakukan langkah 5, 6, 7.
5. Melakukan perhitungan maju dari layar masukan hingga layar keluaran sesuai yang sudah dijelaskan pada bab sebelumnya.
6. Menghitung nilai kesalahan unit *neuron* pada layar keluaran dan layar tersembunyi.
7. Melakukan perubahan pada bobot-bobot yang menghubungkan unit *neuron* layar masukan hingga layar keluaran.
8. Nilai bobot dan bias terakhir setelah pelatihan selesai disimpan sebagai bobot dan bias pada tahap pengujian.

Gambar 3.4 dijelaskan proses implementasi pelatihan BPNN Sesuai dengan diagram alir yang telah diilustrasikan pada Gambar 3.5, maka kode pelatihan BPNN dapat diimplementasikan menggunakan variabel-variabel yang ada pada Tabel 3.3.



Gambar 3.4 Proses Implementasi Pelatihan BPNN

Pada Gambar 3.4 data masukan adalah dua *cluster* terbesar hasil dari proses SOM yang dilakukan sebelumnya. Dua kluster terbesar yang diambil adalah *cluster* yang memiliki proporsi *churn* dan *non-churn* terbanyak. Keluaran dari proses pelatihan adalah nilai bobot dan bias yang akan digunakan pada proses pelatihan BPNN.



Gambar 3.5 Diagram Alir Pelatihan BPNN

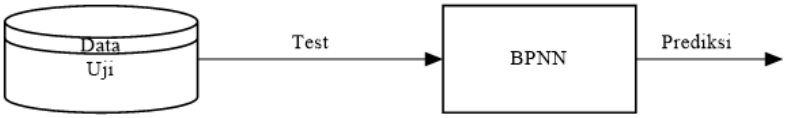
3.3.4.2 Fungsi Pengujian

Pada proses pengujian data masukan adalah *dataset* uji pelanggan. Proses pengujian menggunakan bobot dan bias yang didapatkan dari proses pelatihan. Data keluaran dari proses ini adalah hasil prediksi dan akurasinya. Gambar 3.6 menjelaskan proses implementasi pengujian BPNN. Diagram alir proses pengujian dijelaskan pada Gambar 3.7.

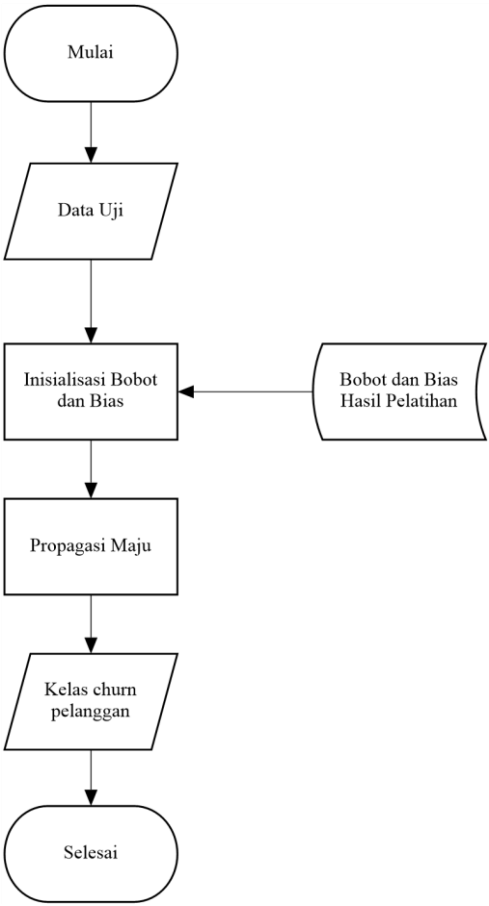
Sesuai dengan diagram alir yang telah diilustrasikan pada Gambar 3.7, maka kode pelatihan BPNN dapat diimplementasikan menggunakan variabel-variabel yang ada pada Tabel 3.4.

Tabel 3.3 Variabel yang Digunakan Pada Pelatihan

No	Variabel	Tipe	Penjelasan
1	num_hidden_layers	int	Jumlah Hidden Layer
2	neuron_on_hidden_layer	int	Jumlah node di hidden layer
3	max_iteration	int	Jumlah Iterasi
4	learning_rate	int	parameter Pembelajaran
5	num_inputs	int	Jumlah Kolom Atribut
6	num_targets	int	Jumlah Kolom Target
7	num_cases	int	Jumlah Data
8	w1	double	Bobot input layer ke hidden layer
9	w2	double	bobot hidden layer ke output layer
10	layers	int	Jumlah layer
11	outputhidden	double	Keluaran neuron di hidden layer
12	outputtarget	double	Keluaran Neuron di Output Layer
13	average_error	double	Rata-rata Error Periterasi



Gambar 3.6 Proses Implementasi Pengujian BPNN



Gambar 3.7 Diagram Alir Pengujian BPNN

Pada tahap pengujian langkah-langkah implementasi pengujian BPNN ditunjukan pada Gambar 3.7. Secara lebih rinci penjelasan pengujian BPNN adalah sebagai berikut:

1. Data masukan tahap pengujian adalah data uji dari *dataset churn* pelanggan

2. Nilai bobot dan bias adalah nilai bobot dan bias hasil dari pelatihan BPNN
3. Proses propagasi maju untuk menentukan nilai kelas target.
4. Hasil akhir dari tahap pengujian adalah kelas target yang didapat melalui proses propagasi maju

Tabel 3.4 Variabel yang Digunakan Pada Pengujian BPNN

No	Variabel	Tipe	Penjelasan
1	num_hidden_layers	int	Jumlah Hidden Layer
2	neuron_on_hidden_layer	int	Jumlah node di hidden layer
3	max_iteration	int	Jumlah Iterasi
4	learning_rate	int	parameter Pembelajaran
5	num_inputs	int	Jumlah Kolom Atribut
6	num_targets	int	Jumlah Kolom Target
7	num_testing_cases	int	Jumlah Data Uji
8	w1	double	Bobot input layer ke hidden layer
9	w2	double	bobot hidden layer ke output layer
10	Layers	int	Jumlah layer
11	outputhidden	double	Keluaran neuron di hidden layer
12	outputtarget	double	Keluaran Neuron di Output Layer
13	average_error	double	Rata-rata Error Periterasi
14	confusion_matrix	double	Konfusion Matrix
15	Accuracy	double	Akurasi Prediksi

Pada Tabel 3.5 menunjukkan *confusion matrix* untuk melakukan perhitungan akurasi. Akurasi didapat menggunakan persamaan 3.2.

$$Acc = \frac{a + d}{a + b + c + d} \tag{3.2}$$

Dimana a adalah *true positive*, b adalah *false negative*, c adalah *false postive*, dan d adalah *true negative*.

Tabel 3.5 *Confusion Matrix*

		Actual	
		Non-churners	*Churners*
Predict	*Non-churners*	*a*	*b*
	Churners	*c*	*d*

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. tahap implementasi dari tiap fungsi akan dijelaskan mengenai parameter masukan, keluaran dan beberapa keterangan yang berhubungan dengan program dan teori.

4.1 Lingkungan Implementasi

Implementasi dilakukan pada lingkungan sebagai berikut:

- Perangkat keras

Uji coba dilakukan pada sebuah PC dengan spesifikasi Processor AMD A6-3650 (2.90 GHz, 8.00 GB RAM), 64-bit Operating System.

- Perangkat lunak

Perangkat lunak ini dikembangkan pada sistem operasi Windows 8 Ultimate dengan menggunakan Matlab 2011a.

4.2 Penjelasan Implementasi

Pada subbab ini akan dijelaskan implementasi setiap subbab yang terdapat pada bab sebelumnya yaitu bab perancangan perangkat lunak, sistem dieksekusi dalam beberapa tahap penting. Terdapat 4 tahap utama antara lain:

1. Normalisasi Data

Seperti yang dijelaskan pada bab sebelumnya pada tahap ini dilakukan perubahan rentang nilai data menggunakan normalisasi *min-max.*

2. Pemisahan Data Uji dan Data latih

Pada tahap ini dilakukan pembagian pada *dataset* untuk menentukan mana yang menjadi data latih dan mana yang menjadi data uji. Terdapat 4 *subset* berbeda yang mempunya proporsi data latih dan data uji yang berbeda.

3. Implementasi SOM

Seperti yang dijelaskan pada bab sebelumnya. Pada tahap implementasi SOM dilakukan pengurangan data latih dengan menggunakan metode *clustering*. Hasil keluaran dari tahap implementasi SOM menjadi data latih pada implementasi JST dengan menggunakan metode *backpropagation.*

4. Implementasi JST

Seperti yang dijelaskan pada bab sebelumnya tahap implementasi JST adalah tahap untuk memprediksi *churn* pelanggan. Pada proses pelatihan data masukan adalah keluaran dari implementasi SOM, dan pada tahap pengujian data masukan adalah data uji dengan nilai bobot dan bias yang didapatkan pada tahap pelatihan.

4.2.1 Normalisasi Data

Sebagaimana dijelaskan pada bab 3 dalam tahap normalisasi data dilakukan pengubahan nilai tiap atribut ke *range* yang sama. Pada tahap ini masukan berubapa *dataset* awal dan keluarannya adalah *dataset* yang telah dinormalisasi *range* nilai atributnya. Kode program ini dapat dilihat pada Kode Sumber 4.1 dan 4.2.

Masukan : *file dataset.*
Keluaran : data yang telah dinormalisasi.

```
1  function[normalized_inputs,normalized_outputs] = normalisasi(input,target)
2   %cek matrix input dan target
3  input_size = size(input);
```

Kode Sumber 4.1 Program Normalisasi Data (Bagian Pertama)

```
4   num_inputs = input_size(2);
5   target_size = size(target);
6   num_targets = target_size(2);
7   assert(input_size(1)==target_size(1)),
8   num_cases = input_size(1);
9   normalized_inputs = input;
10  normalized_outputs = target;
11  min_input = zeros(num_inputs,1);
12  max_input = zeros(num_inputs,1);
13  %normalisasi input
14  for n=1:num_inputs
15      min_input(n) = input(1,n);
16      max_input(n) = input(1,n);
17      for p = 1:num_cases
18          min_input(n) = min(min_input(n), input(p,n));
19          max_input(n) = max(max_input(n), input(p, n));
20      End
21  End
22  %normalisasi output
23  min_output = zeros(num_targets, 1);
24  max_output = zeros(num_targets, 1);
25  for n = 1 : num_targets
33  for p=1:num_cases
34      for n = 1 : num_inputs
35          normalized_inputs(p, n) = (input(p, n) -
    min_input(n)) / (max_input(n) - min_input(n));
36      End
37      for n = 1 : num_targets
```

Kode Sumber 4.2 Program Normalisasi Data (Bagian Kedua)

```
38          normalized_outputs(p, n) = (target(p, n) - min_output(n)) / (max_output(n) - min_output(n));
39      End
40  End
41  End
```

Kode Sumber 4.3 Program Normalisasi Data (Bagian Ketiga)

Pada kode sumber 4.1, 4.2 dan 4.3 baris 35 sampai 39 adalah menghitung nilai atribut baru dengan menggunakan normalisasi *min-max*.

4.2.2 Pemisahan Data Latih dan Data uji

Pada tahap ini dilakukan tahap pemisahan data uji dan data latih untuk menentukan pembagian data mana saja yang menjadi data latih dan data uji. Keluaran dari tahap ini adalah empat buah *subset* yang tiap *subsetnya* memiliki proporsi data latih dan data uji. Kode program ini dapat dilihat pada Kode Sumber 4.4.

Masukan : data, presentase data latih dan data uji
Keluaran : data uji, data latih

```
1  function [training_input_data, training_output_data, testing_input_data, testing_output_data] =
2  traintest(num_cases,num_inputs,num_targets,normalized_inputs, normalized_outputs,x,y)
3  num_training_cases = ceil(x * num_cases);
4  num_testing_cases = ceil(y * num_cases);
5  training_input_data = zeros(num_training_cases, num_inputs);
6  training_output_data = zeros(num_training_cases, num_targets);
7  testing_input_data = zeros(num_testing_cases, num_inputs);
8  testing_output_data = zeros(num_testing_cases, num_targets);
```

Kode Sumber 4.4 Program Pemisahan Data Latih dan Data Uji (Bagian Pertama)

```
9   permutation = randperm(num cases)';
10  %pilih data training
11  for i = 1 : num_training_cases
12      chosen_index = permutation(i);
13      training_input_data(i,           :)          =
    normalized_inputs(chosen_index, :);
14      training_output_data(i,          :)          =
    normalized_outputs(chosen_index, :);
15  End
16  %pilih data testing
17  for i = 1 : num_testing_cases
18      chosen_index = permutation(num_training_cases +
    i);
19      testing_input_data(i,            :)          =
    normalized_inputs(chosen_index, :);
20      testing_output_data(i,           :)          =
    normalized_outputs(chosen_index, :);
21        end
```

Kode Sumber 4.5 Program Pemisahan Data Latih dan Data Uji (Bagian Kedua)

Pada kode sumber 4.4 dan 4.5 dilakukan pembagian data ke data uji dan data latih. Baris 3 menunjukkan perhitungan banyaknya data yang menjadi data latih, baris 4 menunjukkan perhitungan banyaknya data yang menjadi data uji. Baris 10 sampai 21 adalah menentukan indeks data keberapa saja yang menjadi data latih dan data uji.

4.2.3 Implementasi Self-Organizing Map

Seperti yang telah dijelaskan pada bab desain dan perancangan perangkat lunak. Data latih terlebih dahulu melalui proses implementasi SOM. Hasil keluaran dari Implementasi SOM adalah data latih yang telah dikelompokan. Kode sumber 4.6 dan 4.7 menunjukkan implementasi metode SOM.

Masukan : Data latih, data uji, jumlah iterasi,*learning rate* awal, radius awal.

Keluaran : nilai bobot SOM.

```
1  function [som,grid] = som(trainingData, neuronCountW, neuronCountH, trainingSteps, startLearningRate, startRadius)
2  %learning rate awal
3  learning0 = startLearningRate;
4  %radius tetangga awal
5  sigma0 = startRadius;
6  %trainingSteps/log(learning0);
7  T1 = 1000;
8  T2 = 1000/log(sigma0);
9  %insialisasi bobot
10 [datas features] = size(trainingData);
11 totalNeurons = neuronCountW*neuronCountH;
12 som = rand(totalNeurons,features);
13     for i=1:neuronCountH
14         for j=1:neuronCountW
15             pos = ((i-1)*neuronCountW)+(j);
16             grid(pos,:) = [i j];
17         end
18     End
19 %learning rate dan radius sekarang
20 currentLearning = learning0;
21 currentSigma = sigma0;
22     %ulangi sampai batas iterasi
23     for t=1:trainingSteps
24         %masukan vektor input
25         xn = trainingData(randi(datas,1,1),:);
26         %mencari neuron Pemenang
```

Kode Sumber 4.6 Program Utama SOM (Bagian Pertama)

```
27          BMU = findWinnerNeuron(xn,som);
28          for n=1:totalNeurons
29  %Update bobot neuron yang termasuk tetangga dari
    neuron pemenang
30              distanceInfluence(t*n) = exp(-
    ((latticeDistance(grid(BMU,:),grid(n,:))^2)/(2*(curre
    ntSigma(t)^2)) ));
31              som(n,:) = som(n,:) +
    (currentLearning(t)*distanceInfluence(t*n).*(xn-
    som(n,:)));
32          end
33          %Update learning rate
34          currentLearning(t+1) = learning0 * exp(-
    t/T1);
35          %Update Radius
36          currentSigma(t+1) = sigma0 * exp(-t/T2);
37      end
38  End
```

Kode Sumber 4.7 Program Utama SOM (Bagian Kedua)

Pada baris 9 hingga 18 dilakukan insialisasi bobot awal sejumlah total *map output* × jumlah fitur, dan mencari index posisi *neuron* output. Pada baris 30 dan 31 dilakukan pencarian tetangga dari *Best Matching Unit* (BMU). Pada baris 33 hingga 36 dilakukan perubahan nilai *learning rate* dan *radius neighborhood*.

4.2.3.1 Implementasi Fungsi FindWinnerNeuron

Fungsi **FindWinnerNeuron** adalah fungsi untuk mencari neuron output mana yang menjadi pemenang untuk menentukan *kluster* dari data uji. Implementasi fungsi FindWinnerNueron dapat dilihat pada kode sumber 4.8.

Pada fase *clustering* fungsi FindWinnerNeuron dijalankan untuk menentukan termasuk *cluster* mana data tersebut.dengan menggunakan bobot yang dihasilkan dari fase pelatihan.

Masukan : Vektor masukan, bobot.
Keluaran : *neuron* pemenang.

```
1  function winner = findWinnerNeuron (xn,SOM)
2      [numberOfNeurons N] = size(SOM);
3      minMatchingScore = SOM(1,:);
4      winner = 1;
5      for n=1:numberOfNeurons
6          matchingScore = norm(xn-SOM(n,:),2);
7          if matchingScore<minMatchingScore
8              minMatchingScore = matchingScore;
9              winner = n;
```

Kode Sumber 4.8 Fungsi FindWinnerNeuron

Fase *clustering* dilakukan ketika fase pelatihan telah selesai dengan mendapatkan bobot-bobot *node* yang baru. Pada fase ini setelah menerapkan vektor *input* hanya *neuron* pemenang yang dicari.

4.2.3.2 Implementasi Fungsi LatticeDistance

Fungsi Lattice Distance adalah fungsi untuk mencari jarak antara *neuron* pemenang dengan *neuron* lainnya. Implementasi Fungsi LatticeDistance dapat dilihat pada kode sumber 4.9.

Masukan : vektor BMU, Vektor *neuron.*

Keluaran : Jarak.

```
1  Function distance1 = latticeDistance (vect1, vect2)
2  distance1 = norm(vect1 - vect2,1);
3  End
```

Kode Sumber 4.9 Fungsi LatticeDistance

4.2.4 Implementasi Jaringan Saraf Tiruan

Implementasi metode Jaringan Saraf Tiruan menggunakan pelatihan *Backpropagation Neural Network* (BPNN) dilakukan setelah metode SOM dijalankan. Data latih untuk pelatihan BPNN adalah *dataset* hasil dari metode SOM dan data uji untuk metode

pengujian BPNN adalah data uji yang dihasilkan pada tahap pemisahan data uji dan data latih.

```
1  for cv=1:4
2      training_input_data = subset{cv,1};
3      training_output_data = subset{cv,2};
4      testing_input_data = subset{cv,3};
5      testing_output_data = subset{cv,4};
6  %hitung jumlah data training & data testing
7      num_training_cases = size(training_input_data,1);
8      num_testing_cases = size(testing_input_data,1);
9      %insialisasi hidden layer
10     num_hidden_layers=1;
11     neuron_on_hidden_layer=10;
12     %parameter backpropagation
13     max_iteration = 400;
14     learning_rate = 0.05;
15     desired_error = 1e-4
16     %bobot & bias hidden layer
17     for n=1:neuron_on_hidden_layer
18         for i=1:num_inputs
19             w1{i,n} = rand()*2-1;
20             b1{n} = rand()*2-1;
21         end
22     End
17     for n=1:neuron_on_hidden_layer
```

Kode Sumber 4.10 Fungsi Utama BPNN (Bagian Pertama)

```
23  %bobot & bias output layer
24      for n=1:num_targets
25          for i=1:neuron_on_hidden_layer
26              w2{i,n} = rand()*2-1;
27              b2{n} = rand()*2-1;
28          End
28          End
29      End
30      for epoch=1:max_iteration
31          layers = num_hidden_layers+2;
32          %feedforward
33          [outputhidden, outputtarget] = feed(training_input_data,training_output_data,w1,w2,b1,b2,neuron_on_hidden_layer);
34          %mean square error
35          average_error(epoch) = 0;
36          for n=1:num_training_cases
37              sq_sum = 0;
38              for i=1:num_targets
39  delta(n,1) = training_output_data(n,1) - outputtarget(n,1);
40      sq_sum = sq_sum + (delta(n,1)) ^ 2;
41              End
42              error_on_case(n) = 0.5 * sq_sum;
43              average_error(epoch) = average_error(epoch) + error_on_case(n);
44  End
45  average_error(epoch) = average_error(epoch) / num_training_cases;
```

Kode Sumber 4.11 Fungsi Utama BPNN (Bagian Kedua)

```
46  fprintf('On iteration %d average error is %e\n', epoch, average_error(epoch));
47  iterations_made = epoch;
48          if (average_error(epoch) <= desired_error), break; end
49          %backpropagation
50          [errj, errk] = backpropagation(training_input_data,training_output_data,w1,w2,neuron_on_hidden_layer,delta,outputhiddden,outputtarget);
51          %update bobot
52          for i=1:num_targets
53              for j=1:neuron_on_hidden_layer
54                  delta_w2 = learning_rate * (outputhidden'*errj);
55                  w2{j,i} = w2{j,i} + delta_w2(j,i);
56                  b2{i} = b2{i} + learning_rate * sum(outputtarget);
57              End
58          End
59          for i=1:neuron_on_hidden_layer
60              for j=1:num_inputs
61                  delta_w1 = learning_rate*(training_input_data'*errk);
62                  w1{j,i} = w1{j,i} + delta_w1(j,i);
63                  b1{i} = b1{i} + learning_rate * sum(outputhidden(:,i));
64              End
65          End
```

Kode Sumber 4.12 Fungsi Utama BPNN (Bagian Ketiga)

Kode Sumber 4.10, 4.11, dan 4.12 menunjukkan implementasi dari BPNN.

Pada baris 16 hingga 29 dilakukan inisialisasi bobot dan bias pada *hidden layer* dan *output layer*. Nilai bobot dan bias diset *random* kecil dengan *range* nilai antara -1 hingga 1. Pada baris 34

hingga 46 dilakukan penghitungan *mean square error* untuk mengetahui nilai *error* tiap iterasi pelatihan.

4.2.4.1 Fungsi FeedForward

Fungsi Feedforward adalah fungsi untuk melakukan perhitungan maju untuk mendapatkan nilai *neuron* pada tiap layer. Fungsi FeedForward dapat dilihat pada Kode Sumber 4.13, dan 4.14.

Masukan : *input*, *target*, bobot, bias, jumlah *neuron* pada hidden layer.

Keluaran : nilai *neuron* pada *hidden layer* dan *output layer*.

```
1  Function [outputhidden, outputtarget] = feed(input,target,w1,w2,b1,b2,neuron on hidden layer)
2  input_size = size(input);
3  num_inputs = input_size(2);
4  target_size = size(target);
5  num_targets = target_size(2);
6  assert(input_size(1)==target_size(1)),
7  num_cases = input_size(1);
8      for n=1:num_cases
9      %feed forward
10     %input layer ke hidden layer
11         for i=1:neuron_on_hidden_layer
12             totalhidden = 0;
13             for j=1:num_inputs
14                 totalhidden = totalhidden+input(n,j)*w1{j,i};
15             End
16             outputhidden(n,i) = 1/(1+exp(-(totalhidden)));
17 %input hidden layer ke output layer
```

Kode Sumber 4.13 Fungsi FeedForward (Bagian Pertama)

```
18          for i=1:num_targets
19              totaltarget = 0;
20      for j=1:neuron_on_hidden_layer
21                  totaltarget = totaltarget + outputhidden(n,j)*w2{j,i};
22              End
23              outputtarget(n,i) = 1/(1+exp(-(totaltarget)));
24          end
25      End
26  End
```

Kode Sumber 4.14 Fungsi FeedForward (Bagian Kedua)

Pada baris 11 sampai 16 adalah perhitungan nilai *neuron* pada *hidden layer* dengan fungsi aktivasinya pada baris ke-16. Pada baris 17 hingga 23 adalah perhitungan nilai *neuron* pada *output layer* dengan fungsi aktivasinya pada baris ke-23.

4.2.4.2 Fungsi Backpropagation

Fungsi Backpropagation adalah fungsi untuk menghitung kesalahan di *neuron output* dan *neuron* di *hidden layer* yang nantinya akan digunakan untuk perhitungan perubahan bobot. Fungsi Backprop dapat dilihat pada kode sumber 4.15, dan 4.16.

Masukan : *output* data, nilai *neuron* pada *hidden layer* dan *ouput layer*.

Keluaran : *error* pada *neuron output layer* dan *neuron hidden layer*.

```
1   function [errj, errk] = backpropagation(training_input_data,training_output_data,w1,w2,neuron_on_hidden_layer,delta,outputhidden,outputtarget);
2   input_size = size(training_input_data);
```

Kode Sumber 4.15 Fungsi Backproapgation (Bagian Pertama)

```
3   num_inputs = input_size(2);
4   target_size = size(training_output_data);
5   num_targets = target_size(2);
6   assert(input_size(1)==target_size(1)),
7   num_cases = input_size(1);
8       for n=1:num_cases
9           %output node
10          for i=1:num_targets
11  errj(n,i)           =           outputtarget(n,i)*(1-
    outputtarget(n,i))*delta(n,1);
12          End
13          %hidden node
15              for j=1:num_targets
16              errk(n,i)     =     outputhidden(n,i)*(1-
    outputhidden(n,i))*(errj(n,j)*w2{i,j});
17              End
18          End
19      End
20  End
```

Kode Sumber 4.16 Fungsi Backproapgation (Bagian Kedua)

Pada baris 9 sampai 12 dilakukan perhitungan *error neuron* di *output layer*. Baris 13 sampai 17 adalah perhitungan *error neuron* di *hidden layer*.

4.2.4.3 Fungsi WeightUpdate

Fungsi WeightUpdate adalah fungsi untuk mengubah bobot yang menghubungkan antara *neuron* masukan ke *neuron* di *hidden layer* dan *neuron* di *hidden layer* ke neuron di *output layer*. Fungsi WeightUpdate dapat dilihat pada kode sumber 4.17

Masukan : *learning rate*, *errj*, *errk*.
Keluaran : Nilai bobot baru.

```
1  %update bobot
2  for i=1:num_targets
3      for j=1:neuron_on_hidden_layer
4          Delta_w2 = learning_rate * (outputhidden'*errj);
5          w2{j,i} = w2{j,i} + delta_w2(j,i);
6          b2{i} = b2{i} + learning_rate * sum(outputtarget);
7                  End
8        end
9            for i=1:neuron_on_hidden_layer
10                 for j=1:num_inputs
11                     delta_w1 = learning_rate*(training_input_data'*errk);
12                     w1{j,i} = w1{j,i} + delta_w1(j,i);
13                     b1{i} = b1{i} + learning_rate * sum(outputhidden(:,i));
14                 End
15             End
16       end
```

Kode Sumber 4.17 WeightUpdate

Pada baris 3 sampai 8 dilakukan perhitungan perubahan bobot yang menghubungkan *neuron hidden layer* ke *neuron output layer*. Baris 9 sampai 15 menghitung perubahan bobot yang menghubungkan *neuron input* ke *neuron hidden layer*.

4.2.4.4 Fungsi Pengujian BPNN

Pada tahap ini dilakukan pengujian terhadap data uji menggunakan BPNN. Pada tahap ini data masukan adalah data uji dengan bobot antar *neuron* adalah hasil bobot yang didapatkan pada pelatihan BPNN. Hasil akurasi BPNN dihitung menggunakan *confusion matrix*. Kode sumber 4.18 adalah fungsi pengujian BPNN.

Masukan : Data uji, bobot, bias, target, *neuron* di *hidden layer*.
Keluaran : nilai *neuron hidden layer* dan *output layer*, Akurasi.

```
1  %feed total data
2  [total_output_hidden ,total_output] = feed(testing_input_data,testing_output_data,w1,w2,b1, b2,neuron_on_hidden_layer);
3  total_average_error = 0;
4     for p = 1 : num_testing_cases
5           sq_sum = 0;
6           for i = 1 : num_targets
7              sq_sum = sq_sum + (target(p, i) - total_output(p, i)) ^ 2;
8           end
9           total_average_error = total_average_error + 0.5 * sq_sum;
10       end
11 total_average_error =total_average_error / num_testing_cases;
12 fprintf('Average error on total data is %e\n', total_average_error);
13 [confusion_matrix2,order]= confusionmat(zzz,testing_output_data, 'order', order);
14 [confusion_matrix2,order]= confusionmat(zzz,testing_output_data, 'order', order);
15 total_accuracy = ((confusion_matrix2(1,1)+confusion_matrix2(2,2))/(con fusion_matrix2(1,1)
16 +confusion_matrix2(1,2)+confusion_matrix2(2,1)+confus ion_matrix2(2,2)))*100;
17 accuracy(cv) = total_accuracy;
```

Kode Sumber 4.18 Fungsi Pengjian BPNN

Proses pengujian adalah melakukan tahap propagasi maju dengan menggunakan nilai bobot dan bias yang didapat dari proses pelatihan BPNN. Baris 2 menunjukkan fungsi FeedForward dengan keluarannya adalah nilai dari *neuron* pada *hidden layer* dan nilai *neuron* pada *output layer*. Baris 1 sampai 11 menunjukkan perhitungan *mean square error* pada proses pengujian BPNN dan

baris 11 sampai 17 menunjukkan penghitungan akurasi dari pengujian.

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan terhadap sistem yang dibuat. Pembahasan yang dipaparkan meliputi lingkungan uji coba, data uji coba, skenario uji coba, hasil uji coba, dan evaluasi.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada lingkungan sebagai berikut:

- Perangkat keras

Uji coba dilakukan pada sebuah PC dengan spesifikasi Processor AMD A6-3650 (2.90 GHz, 8.00 GB RAM), 64-bit Operating System.

- Perangkat lunak

Perangkat lunak ini dikembangkan pada sistem operasi Windows 8 Ultimate dengan menggunakan Matlab 2011a.

5.2 Data Uji Coba

Data uji coba yang digunakan adalah *dataset* pelanggan yang didapat dari PT.Telkomsel. Jumlah data sebanyak 1000 data dengan jumlah atribut sebanyak 64 dengan 1 atribut target. Class target padata *dataset* bernilai 1 dan 0 dengan 1 menandakan *churn* dan 0 menandakan *non-churn*. Pelanggan yang termasuk dalam *dataset* ini adalah pelanggan yang masa berlaku kartunya tinggal 20 hari lagi. Tabel 5.1, 5.2, 5.3, dan 5.4 menjelaskan atribut pada *dataset* pelanggan dan contoh *field*-nya.

Tabel 5.1 Informasi Atribut dataset Pelanggan (Bagian Pertama)

Nomor	Atribut	Keterangan
1	RECH_30DAYS	1000,10000,20000
2	RECH_45DAYS	1000,10000,20000
3	RECH_60DAYS	1000,10000,20000
4	RECH_75DAYS	1000,10000,20000
5	RECH_90DAYS	1000,10000,20000
6	GROWTH_RECH_3045	-0.9,-0.8,-0.5,0,1,1.5,1.9
7	GROWTH_RECH_4560	-0.9,-0.8,-0.5,0,1,1.5,1.9
8	GROWTH_RECH_6075	-0.9,-0.8,0,0.5,1,1.5
9	GROWTH_RECH_7590	-0.9,-0.8,0,0.5,1,1.5
10	REV_VOICE_30DAYS	1000,10000,20000
11	REV_VOICE_45DAYS	1000,10000,20000
12	REV_VOICE_60DAYS	1000,10000,20000
13	REV_VOICE_75DAYS	1000,10000,20000
14	REV_VOICE_90DAYS	1000,10000,20000
15	GROWTH_REV_VOICE_3045	-1,-0.9,-0.8,1.25,7.66
16	GROWTH_REV_VOICE_4560	-1,-0.9,-0.8,1.25,7.66
17	GROWTH_REV_VOICE_6075	-1,-0.9,-0.8,1.25,7.66
18	GROWTH_REV_VOICE_7590	-1,-0.9,-0.8,1.25,7.66
19	MOU_VOICE_30DAYS	0.05,73.56,120,180
20	MOU_VOICE_45DAYS	0.05,73.56,120,180
21	MOU_VOICE_60DAYS	0.05,73.56,120,180

Tabel 5.2 Informasi Atribut *dataset* Pelanggan (Bagian Kedua)

22	MOU_VOICE_75DAYS	0.05,73.56,120,180
23	MOU_VOICE_90DAYS	0.05,73.56,120,180
24	GROWTH_MOU_VOICE_3045	-0.9,-0.8,-0.23,1.4,1.9
25	GROWTH_MOU_VOICE_4560	-0.9,-0.8,-0.23,1.4,1.9
26	GROWTH_MOU_VOICE_6075	-0.9,-0.8,-0.23,1.4
27	GROWTH_MOU_VOICE_7590	-0.9,-0.8,-0.23,1.4
28	TRX_VOICE_30DAYS	28
29	TRX_VOICE_45DAYS	28
30	TRX_VOICE_60DAYS	28
31	TRX_VOICE_75DAYS	28
32	TRX_VOICE_90DAYS	28
33	GROWTH_TRX_VOICE_3045	1
34	GROWTH_TRX_VOICE_4560	1
35	GROWTH_TRX_VOICE_6075	1
36	GROWTH_TRX_VOICE_7590	1
37	REV_SMS_30DAYS	100000
38	REV_SMS_45DAYS	100000
39	REV_SMS_60DAYS	100000
40	REV_SMS_75DAYS	100000
41	REV_SMS_90DAYS	100000
42	GROWTH_REV_SMS_3045	1
43	GROWTH_REV_SMS_4560	1
44	GROWTH_REV_SMS_6075	1
45	GROWTH_REV_SMS_7590	1
46	TRX_SMS_30DAYS	100000
47	TRX_SMS_45DAYS	100000

Tabel 5.3 Informasi Atribut *dataset* Pelanggan (Bagian Ketiga)

48	TRX_SMS_60DAYS	100000
49	TRX_SMS_75DAYS	100000
50	TRX_SMS_90DAYS	100000
51	GROWTH_TRX_SMS_3045	1
52	GROWTH_TRX_SMS_4560	1
53	GROWTH_TRX_SMS_6075	1
54	GROWTH_TRX_SMS_7590	1
55	REV_DATA_30DAYS	100000
56	REV_DATA_45DAYS	100000
57	REV_DATA_60DAYS	100000
58	REV_DATA_75DAYS	100000
59	REV_DATA_90DAYS	100000
60	GROWTH_REV_DATA_3045	1
61	GROWTH_REV_DATA_4560	1
62	GROWTH_REV_DATA_6075	1
63	GROWTH_REV_DATA_7590	1
64	Class	*Churn*/*non-churn*

5.3 Skenario & Hasil Uji Coba

Pengujian dilakukan dengan menggunakan beberapa metode sesuai dengan algoritma yang telah dijelaskan pada bab-bab sebelumnya. Skenario pengujian utama adalah pengaplikasian metode BPPN untuk prediksi *churn* pelanggan yang digabung dengan metode SOM sebagai metode *cluster* untuk pengurangan data yang akan digunakan sebagai data latih BPNN. Skenario pengujian dibandingkan dengan skenario pengaplikasian metode

untuk prediksi *churn* pelanggan yang tidak digabung dengan metode SOM.

Perlu dilakukan proses normalisasi data yang bertujuan agar semua atribut mempunyai rentang nilai yang sama. Normalisasi mencegah atribut yang memiliki rentang data luas bernilai lebih besar daripada atribut dengan rentang data kecil.

Setiap uji coba memiliki data latih dan data uji yang berbeda. Terdapat empat *subset* yang menunjukkan proporsi data latih dan data uji yang akan digunakan pada tiap uji coba. Tabel 5.4 menunjukkan proporsi dari tiap *subset*.

Tabel 5.4 *Subset*

Subset	Data Latih	Data Uji
Subset 1	60%	40%
Subset 2	70%	30%
Subset 3	80%	20%
Subset 4	90%	10%

5.3.1 Uji Coba JST

Uji coba JST merupakan skenario untuk mengetahui akurasi prediksi *churn* dengan menggunakan dataset pelanggan. Uji coba ini dilakukan tanpa penggunaan metode SOM sehingga jumlah data tidak mengalami pengurangan. Pada uji coba ini digunakan empat iterasi pembelajaran (400, 450, 475, 500) dan 5 jenis *node* di hidden layer (8, 12, 16, 24, dan 32) untuk mendapatkan hasil terbaik. Terdapat 20 jenis model BPNN berbeda yang didapat.

5.3.1.1 Uji Coba Pertama

Pada Tabel 5.5 adalah uji coba pertama dengan jumlah iterasi pelatihan adalah 400 kali dengan menggunakan empat *subset* yang mewakili proporsi data latih dan data uji yang berbeda.

Pada uji coba pertama yang ditunjukan oleh Tabel 5.5. Rata-rata akurasi yang memiliki hasil terbaik adalah model BPNN

dengan jumlah *neuron* di *hidden layer* sebanyak 12 neuron. Selain itu *subset* yang memiliki akurasi tertinggi adalah *subset* pertama yang memiliki akurasi sebesar 94.75% dengan jumlah *neuron* di *hidden layer* sebanyak 8 *neuron*.

Tabel 5.5 Hasil Akurasi Uji Coba JST Pertama

Hidden Nodes	8	10	12	14	16
Subset 1	**94.75**	92.25	93.25	92.00	93.00
Subset 2	90.75	90.25	93.50	92.50	90.00
Subset 3	93.75	91.50	93.75	91.25	92.75
Subset 4	91.25	91.75	91.75	90.75	91.00
Average	92.62	91.43	**93.06**	91.62	91.8

5.3.1.2 Uji Coba Kedua

Uji Coba kedua memiliki jumlah pelatihan sebanyak 450 kali dengan menggunakan empat *subset* yang mewakili proporsi jumlah data pelatihan dan data uji yang berbeda. Tabel 5.6 menunjukkan hasil uji coba BPNN yang kedua.

Tabel 5.6 Hasil Akurasi Uji Coba JST Kedua

Hidden Nodes	8	10	12	14	16
Subset 1	93.75	93.50	92.50	92.50	92.00
Subset 2	93.00	91.75	93.50	89.00	**94.25**
Subset 3	93.25	**94.25**	92.75	91.75	91.00
Subset 4	93.00	94.00	90.00	91.75	91.00
Average	93.25	**93.37**	92.1875	91.25	92.06

Pada hasil uji coba kedua yang ditunjukan oleh Tabel 5.6. Rata-rata akurasi yang memiliki hasil terbaik adalah model BPNN dengan jumlah *neuron* di *hidden layer* sebanyak 10 *neuron* akurasi yang dihasilkan sebesar 93.37%. *Subset* yang memilki akurasi tertinggi adalah *subset* ke-2 dengan jumlah *neuron* di *hidden layer* sebanyak 16 *neuron* yang menghasilkan akurasi sebesar 94.25%,

dan *subset* ke-3 dengan jumlah *neuron* di *hidden layer* sebanyak 10 *neuron* yang menghasilkan akurasi sebesar 94.25%.

5.3.1.3 Uji Coba Ketiga

Uji coba ketiga memiliki jumlah pelatihan sebanyak 475 kali dengan menggunakan empat *subset* yang mewakili proporsi jumlah data pelatihan dan data pengujian. Tabel 5.7 menunjukkan hasil uji coba BPNN ketiga.

Tabel 5.7 Hasil Akurasi Uji Coba JST Ketiga

Hidden Nodes	8	10	12	14	16
Subset 1	92.75	93.75	91.50	93.00	**94.75**
Subset 2	91.75	94.00	93.00	93.50	89.25
Subset 3	93.50	93.75	93.00	92.75	94.00
Subset 4	93.00	94.00	87.25	90.75	92.50
Average	**92.75**	93.87	91.18	92.50	92.62

Pada hasil uji coba ketiga yang ditunjukan oleh Tabel 5.7 hasil akurasi yang memiliki rata-rata paling tinggi dimiliki model BPNN yang mempunyai 8 *neuron* di *hidden layer* dengan akurasi sebesar 92.75%. *Subset* yang menghasilkan akurasi tertinggi adalah *subset* ke-1 dengan nilai akurasi sebesar 94.75% dengan jumlah *neuron* di *hidden layer* berjumlah 16 *neuron*.

5.3.1.4 Uji Coba Keempat

Uji Coba keempat memiliki jumlah pelatihan sebanyak 500 kali iterasi dengan menggunakan data masukan empat *subset* yang mewakili proporsi jumlah data pelatihan dan data pengujian. Tabel 5.8 menunjukkan tabel hasil uji coba model BPNN dengan jumlah iterasi 500 kali pelatihan.

Pada hasil uji coba BPNN keempat rata-rata akurasi tertinggi yang didapat adalah 92.12% dengan jumlah *neuron* di *hidden layer* adalah 8 dan 10. Subset yang menghasilkan akurasi tertinggi adalah subset ke-2 dengan nilai akurasi sebesar 94.25% dengan jumlah *neuron* pada *hidden layer* sebanyak 12 *neuron*.

Tabel 5.8 Hasil Akurasi Uji Coba JST Keempat

Hidden Nodes	8	10	12	14	16
Subset 1	92.75	89.25	88.75	92.75	93.75
Subset 2	92.50	94.00	**94.25**	88.00	88.50
Subset 3	91.25	95.25	93.00	93.00	92.50
Subset 4	92.00	90.00	88.75	88.00	88.50
Average	**92.12**	**92.12**	91.18	90.43	90.81

5.3.2 Uji Coba JST-SOM

Uji Coba JST-SOM merupakan skenario untuk mengetahui hasil akurasi prediksi *churn* pelanggan. Skenario ini menggabungkan metode JST dan SOM. SOM digunakan untuk pengurangan data latih dengan melakukan pengelompokan data yang mempunyai pola yang mirip. Hasil dari pengelompokkan menggunakan SOM akan digunakan sebagai data latih metode JST menggunakan *Backpropagation Neural Network* (BPNN). Besar SOM di set memiliki ukuran 2x2 sehingga menghasilkan empat jenis kluster. Dua kluster terbesar hasil SOM yang mempunyai proporsi pelanggan yang *churn* dan tidak *churn* dipilih untuk menjadi data uji BPNN. Tabel 5.9, 5.10, 5.11, 5.12 menunjukkan hasil *cluster* tiap subset dari SOM.

Tabel 5.9 adalah hasil *cluster* yang terbentuk pada *subset* 1. Dua *cluster* terbesar pada *subset* 1 adalah *cluster* 1 dan 3 dengan jumlah total pelanggan yang *churn* adalah 430 dan pelanggan yang *non-churn* adalah 58. Hasil *cluster* 1 dan 3 akan dijadikan sebagai data latih pada JST.

Tabel 5.10 adalah hasil *cluster* yang terbentuk pada *subset* 2. Dua *cluster* terbesar pada *subset* 1 adalah *cluster* 1 dan 4 dengan jumlah total pelanggan yang *churn* adalah 479 dan pelanggan yang *non-churn* adalah 51. Hasil *cluster* 1 dan 4 akan dijadikan sebagai data latih *subset* 2 pada JST.

Tabel 5.11 adalah hasil *cluster* yang terbentuk pada *subset* 3. Dua *cluster* terbesar pada *subset* 1 adalah *cluster* 2 dan 3 dengan jumlah total pelanggan yang *churn* adalah 547 dan pelanggan yang

non-churn adalah 55. Hasil *cluster* 2 dan 3 akan dijadikan sebagai data latih *subset* 3 pada JST.

Tabel 5.9 *Cluster* SOM *Subset* 1

Cluster	*Class*		Total
	0	1	
1	38	323	361
2	0	3	3
3	20	107	127
4	6	103	109
	64	536	600

Tabel 5.10 *Cluster* SOM *Subset* 2

Cluster	*Class*		Total
	0	1	
1	14	158	172
2	16	125	141
3	4	25	29
4	37	321	358
	71	629	700

Tabel 5.12 adalah hasil *cluster* yang terbentuk pada *subset* 4. Dua *cluster* terbesar pada *subset* 1 adalah *cluster* 1 dan 3 dengan jumlah total pelanggan yang *churn* adalah 717 dan pelanggan yang *non-churn* adalah 93. Hasil *cluster* 1 dan 3 akan dijadikan sebagai data latih *subset* 4 pada JST.

Pada pengujian JST-SOM dibentuk 20 model berbeda dengan jumlah iterasi sebanyak (400, 450, 475, 500) dan jumlah *neuron* pada hidden layer (8, 10, 12, 14, 16). Terdapat empat *subset* yang mewakili proporsi data uji dan data latih yang berbeda antara satu dan lainnya.

Tabel 5.11 *Cluster* SOM *Subset* 3

Cluster	*Class*		Total
	0	1	
1	34	119	153
2	38	374	412
3	17	173	190
4	2	43	45
	91	709	800

Tabel 5.12 *Cluster* SOM *Subset* 4

Cluster	*Class*		Total
	0	1	
1	16	173	189
2	5	77	82
3	77	544	621
4	0	8	8
	98	802	900

5.3.2.1 Uji Coba Pertama

Pada uji coba pertama dilakukan pelatihan sebanyak 400 kali dengan empat *subset* yang mempunyai proporsi data latih dan data uji berbeda. Sebelum dilakukan pelatihan menggunakan BPNN dilakukan dulu implementasi SOM dengan data masukan yaitu data latih yang didapat dari tiap *subset*, lalu hasil dari SOM menjadi data latih pada BPNN. Tabel 5.13 menunjukkan uji coba JST-SOM pertama.

Pada uji coba JST-SOM pertama rata-rata akurasi tertinggi yang didapat adalah 95.43% yang dihasilkan dari model dengan jumlah *neuron* pada *hidden neuron* adalah 8. *Subset* yang menghasilkan akurasi tertinggi adalah *subset* kedua dengan akurasi

sebesar 95.9% dengan model yang mempunyai *neuron* pada *hidden layer* berjumlah 10 *neuron*.

Tabel 5.13 Hasil Akurasi Uji Coba JST-SOM Pertama

Hidden Nodes	8	10	12	14	16
Subset 1	95.75	95.75	96.00	95.75	95.25
Subset 2	95.50	**95.90**	94.50	95.80	95.30
Subset 3	95.30	93.20	94.30	93.00	95.00
Subset 4	95.20	95.60	95.60	95.10	94.70
Average	**95.43**	95.11	95.10	94.91	95.06

5.3.2.2 Uji Coba Kedua

Uji coba kedua dilakukan pelatihan sebanyak 450 kali dengan empat *subset* yang mempunyai proporsi data latih dan data uji berbeda. Sebelum dilakukan pelatihan menggunakan BPNN dilakukan dulu implementasi SOM dengan data masukan yaitu data latih yang didapat dari tiap *subset*, lalu hasil dari SOM menjadi data latih pada BPNN.

Tabel 5.14 Hasil Akurasi Uji Coba JST-SOM Kedua

Hidden Nodes	8	10	12	14	16
Subset 1	94.75	95.25	94.50	94.00	**9**6.00
Subset 2	96.30	95.80	96.10	**96.40**	94.20
Subset 3	95.20	95.20	94.40	94.90	94.00
Subset 4	94.60	95.60	95.80	95.60	95.80
Average	95.21	95.21	95.20	**95.22**	94.75

Tabel 5.14 menunjukkan uji coba JST-SOM kedua. Pada uji coba JST-SOM pertama rata-rata akurasi tertinggi yang didapat adalah 95.22% yang dihasilkan dari model dengan jumlah *neuron* pada *hidden neuron* adalah 14. *Subset* yang menghasilkan akurasi tertinggi adalah *subset* kedua dengan akurasi sebesar 96.4% dengan

model yang mempunyai *neuron* pada *hidden layer* berjumlah 14 *neuron*.

5.3.2.3 Uji Coba Ketiga

Uji coba ketiga dilakukan pelatihan sebanyak 475 kali dengan empat *subset* yang mempunyai proporsi data latih dan data uji berbeda. Sebelum dilakukan pelatihan menggunakan BPNN dilakukan dulu implementasi SOM dengan data masukan yaitu data latih yang didapat dari tiap *subset*, lalu hasil dari SOM menjadi data latih pada BPNN. Tabel 5.15 menunjukkan uji coba JST-SOM ketiga.

Pada uji coba JST-SOM ketiga rata-rata akurasi tertinggi yang didapat adalah 95.41% yang dihasilkan dari model dengan jumlah *neuron* pada *hidden layer* adalah 14. *Subset* yang menghasilkan akurasi tertinggi adalah *subset* keempat dengan akurasi sebesar 96.7% dengan model yang mempunyai *neuron* pada *hidden layer* berjumlah 12 *neuron*.

Tabel 5.15 Hasil Akurasi Uji Coba JST-SOM Ketiga

Hidden Nodes	8	10	12	14	16
Subset 1	94.50	94.75	94.50	94.25	94.50
Subset 2	95.30	96.30	95.60	95.10	94.80
Subset 3	94.60	94.60	94.90	96.10	94.90
Subset 4	95.20	95.60	**96.70**	96.20	95.60
Average	94.90	95.31	95.42	**95.41**	94.95

5.3.2.4 Uji Coba Keempat

Uji coba keempat dilakukan pelatihan sebanyak 500 kali dengan empat *subset* yang mempunyai proporsi data latih dan data uji berbeda. Sebelum dilakukan pelatihan menggunakan BPNN dilakukan dulu implementasi SOM dengan data masukan yaitu data latih yang didapat dari tiap *subset*, lalu hasil dari SOM menjadi data latih pada BPNN. Tabel 5.16 menunjukkan uji coba JST-SOM kedua.

Pada uji coba JST-SOM keempat rata-rata akurasi tertinggi yang didapat adalah 95.68% yang dihasilkan dari model dengan jumlah *neuron* pada *hidden layer* adalah 8. *Subset* yang menghasilkan akurasi tertinggi adalah *subset* keempat dengan akurasi sebesar 96.6% dengan model yang mempunyai *neuron* pada *hidden layer* berjumlah 8 *neuron*.

Tabel 5.16 Hasil Akurasi Uji Coba JST-SOM Keempat

Hidden Nodes	8	10	12	14	16
Subset 1	94.75	94.50	94.25	94.25	94.75
Subset 2	95.60	95.90	95.50	95.10	93.90
Subset 3	95.80	95.80	94.60	95.80	95.10
Subset 4	**96.60**	95.10	96.30	96.30	96.10
Average	**95.68**	95.07	95.16	95.36	94.96

5.4 Evaluasi

Uji Coba dilakukan sebanyak 8 skenario dengan empat skenario menggunakan penggabungan JST-SOM dan empat skenario menggunakan metode JST saja. Tiap skenario memiliki jumlah iterasi pembelajaran yang berbada dan jumlah *neuron* pada *hidden layer* juga berbeda satu dengan yang lain. Pada 8 skenario percobaan ini didapat beberapa kesimpulan.

Evaluasi uji coba JST menampilkan akurasi tertinggi tiap *subset*. Setiap model dibandingkan untuk mencari nilai akurasi paling tinggi pada tiap subset, terdapat 20 model yang terbentuk dalam uji coba sebelumnya dengan jumlah iterasi (400, 450, 475, 500) dan jumlah *neuron* pada *hidden layer* (8, 10, 12, 14, 16). Tabel 5.17 menunjukkan hasil akurasi tertinggi tiap subset pada 20 model BPNN yang diuji.

Evaluasi uji coba JST-SOM menampilkan akurasi tertinggi tiap *subset*. Setiap model dibandingkan untuk mencari nilai akurasi paling tinggi pada tiap *subset* terdapat 20 model yang terbentuk dalam uji coba sebelumnya dengan jumlah iterasi (400, 450, 475,

500) dan jumlah *neuron* pada *hidden layer* (8, 10, 12, 14, 16). Tabel 5.17 menunjukkan hasil akurasi tertinggi tiap *subset* pada 20 model JST. Tabel 5.18 menunjukkan hasil akurasi tertinggi tiap *subset* pada 20 model JST-SOM yang diuji.

Tabel 5.17 Hasil Akurasi Prediksi JST

Testing Set	*Learning Epoch*	*Hidden Nodes*	Akurasi
Subset 1	400	8	94.75
Subset 2	450	16	94.25
Subset 3	500	10	95.25
Subset 4	475	10	94.00

Tabel 5.18 Hasil Akurasi Prediksi JST-SOM

Testing Set	*Learning Epoch*	*Hidden Nodes*	Akurasi
Subset 1	450	16	96.00
Subset 2	450	14	96.40
Subset 3	475	14	96.10
Subset 4	475	12	96.70

Secara umum tidak ada ada perbedaan jauh antara akurasi *subset* satu dan lainnya. Pada uji coba JST-SOM akurasi paling tinggi yang didapat adalah 96.7% dengan jumlah iterasi pelatihan adalah 475 dan *neuron* pada *hidden layer* berjumlah 12. Sedangkan akurasi JST paling tinggi adalah 94.75% sehingga dapat ditarik kesimpulan terjadi kenaikan akurasi sebesar1.95% JST apabila digabungkan SOM.

BAB VI
KESIMPULAN DAN SARAN

Dalam bab terakhir ini dijelaskan kesimpulan yang didapat dari pengerjaan Tugas Akhir beserta saran-saran yang dapat dipertimbangkan untuk pengembangan atau penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap pembuatan program implementasi metode hybrid JST-SOM untuk prediksi *churn* dapat diambil kesimpulan sebagai berikut:

1. Penggabungan dua metode *data mining* yaitu *classification* dan *clustering* dapat dilakukan untuk meningkatkan hasil akurasi sebuah model.
2. Akurasi yang dihasilkan oleh JST yang terlebih dahulu melalui proses implementasi SOM mempunyai akurasi yang lebih tinggi daripada tanpa dilakukan implementasi SOM terlebih dahulu dengan peningkatan akurasi sebesar 1.95% dan dengan nilai akurasi tertinggi adalah 96.70%.
3. Tidak ada perbedaan akurasi yang signifikan terhadap banyaknya jumlah *neuron* pada *hidden layer.*
4. Tidak ada perbedaan akurasi yang signifikan terhadap banyaknya jumlah iterasi pelatihan.

6.2 Saran

Saran yang diberikan untuk pengembangan Implementasi Metode *Hybrid* JST-SOM pada Tugas Akhir ini antara lain:

1. Diperlukan proses pemilihan atribut lain yang mungkin dapat meningkatkan akurasi prediksi *churn* pelanggan
2. Perlu diperlukan uji coba dengan jumlah *hidden layer* lebih dari satu yang mungkin dapat meningkatkan akurasi prediksi *churn* pelanggan.

3. Perlu dibuat antarmuka yang baik sehingga dapat dioperaskan dengan mudah.
4. Perlu uji coba dengan metode *clustering* lain contohnya k-means untuk membandingkan dengan JST-SOM.

DAFTAR PUSTAKA

[1] C.-F. Tsai and Y.-H. Lu, "Customer churn prediction by hybrid neural networks," *Expert Systems with Applications*, p. 12547–12553, 2009.

[2] M. T. K. a. B. B. Binquan Huang, "Customer Churn Prediction in Telecommunications," *Expert System with Applications*, vol. 39, pp. 1414-1425, 2012.

[3] P. N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, 4th ed., Boston: Pearson Addison Wesley, 2006.

[4] S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Third Edition, New Jersey: Pearson, 2010.

[5] L. V. Fausett, Fundamental of neural networks: architecture, algorithms, and applications, 1994.

[6] C. Hung and C.-F. Tsai, "Segmentation based on hierarchical self-organizing map for markets of multimedia on demand," *Expert Systems with Applications*, vol. 34(1), p. 780–787, 2008.

LAMPIRAN

A. Hasil Lengkap Uji Coba

Pada bagian lampiran ini berisi hasil-hasil percobaan secara keseluruhan. Tabel berisi hasil percobaan JST-SOM. Percobaan dilakukan sebanyak 10 kali tiap jumlah node pada hidden layer. Terdapat 20 jenis model dengan jumlah iterasi dan node pada *hidden layer* yang berbeda antara satu dan lainnya.

Tabel A.1 Hasil Uji Coba JST-SOM *Subset* 1 (Bagian Pertama)

No	Node Hidden	Iterasi	Akurasi
1	8	400	94.00
2	8	400	94.50
3	8	400	94.00
4	8	400	94.50
5	8	400	95.00
6	8	400	95.00
7	8	400	95.50
8	8	400	95.75
9	8	400	95.75
10	8	400	95.75
11	10	400	95.00
12	10	400	94.50
13	10	400	94.50
14	10	400	95.00
15	10	400	94.75
16	10	400	94.75
17	10	400	95.25

Tabel A.2 Hasil Uji Coba JST-SOM Subset 1 (Bagian Kedua)

18	10	400	95.75
19	10	400	95.75
20	10	400	95.75
21	12	400	92.75
22	12	400	94.25
23	12	400	94.75
24	12	400	94.25
25	12	400	95.25
26	12	400	95.50
27	12	400	95.50
28	12	400	95.50
29	12	400	96.00
30	12	400	96.00
31	14	400	95.25
32	14	400	95.00
33	14	400	95.00
34	14	400	95.50
35	14	400	95.50
36	14	400	95.50
37	14	400	95.50
38	14	400	95.75
39	14	400	95.75
40	14	400	95.25
41	16	400	95.25
42	16	400	95.00
43	16	400	95.00

Tabel A.3 Hasil Uji Coba JST-SOM Subset 1 (Bagian Ketiga)

45	16	400	95.25
46	16	400	95.25
47	16	400	94.75
48	16	400	95.25
49	16	400	95.00
50	16	400	94.00
51	8	450	94.75
52	8	450	94.75
53	8	450	94.00
54	8	450	94.75
55	8	450	94.75
56	8	450	94.75
57	8	450	94.50
58	8	450	94.75
59	8	450	94.00
60	8	450	94.25
61	10	450	94.25
62	10	450	94.00
63	10	450	94.50
64	10	450	94.50
65	10	450	94.25
66	10	450	94.50
67	10	450	94.50
68	10	450	94.25
69	10	450	94.00
70	10	450	94.75

Tabel A.4 Uji Coba JST-SOM Subset 1
(Bagian Keempat)

71	12	450	94.00
72	12	450	94.50
73	12	450	94.00
74	12	450	94.50
75	12	450	93.75
76	12	450	94.50
77	12	450	94.00
78	12	450	94.25
79	12	450	94.25
80	12	450	94.50
81	14	450	93.50
82	14	450	93.25
83	14	450	94.00
84	14	450	93.25
86	14	450	93.25
87	14	450	93.75
88	14	450	93.50
89	14	450	94.00
90	14	450	95.00
91	16	450	95.00
92	16	450	94.25
93	16	450	94.50
94	16	450	94.25
95	16	450	95.00
96	16	450	94.50
97	16	450	94.50
98	16	450	94.75

Tabel A.5 Uji Coba JST-SOM Subset 1 (Bagian Kelima)

99	16	450	95.00
100	16	450	95.00
101	8	475	94.50
102	8	475	94.00
103	8	475	94.50
104	8	475	93.75
105	8	475	93.75
106	8	475	94.00
107	8	475	94.25
108	8	475	94.50
109	8	475	94.25
110	8	475	94.00
111	10	475	94.50
112	10	475	94.50
113	10	475	94.75
114	10	475	94.66
115	10	475	94.00
116	10	475	94.33
117	10	475	94.33
118	10	475	94.00
119	10	475	94.00
120	10	475	94.66
121	12	475	94.00
122	12	475	94.33
123	12	475	94.00
124	12	475	94.50
125	12	475	94.50

Tabel A.6 Uji Coba JST-SOM Subset 1 (Bagian keenam)

126	12	475	94.50
127	12	475	94.00
128	12	475	94.33
129	12	475	94.00
130	12	475	94.50
131	14	475	94.25
132	14	475	94.25
133	14	475	94.33
134	14	475	94.25
135	14	475	94.00
136	14	475	94.00
137	14	475	94.00
138	14	475	94.00
139	14	475	94.00
140	14	475	94.00
141	16	475	94.50
142	16	475	94.00
143	16	475	93.66
144	16	475	94.50
145	16	475	94.50
146	16	475	94.50
147	16	475	94.00
148	16	475	94.00
149	16	475	94.50
150	16	475	93.33
151	8	500	94.00
152	8	500	94.25
153	8	500	94.75

Tabel A.7 Hasil Uji Coba JST-SOM Subset 1 (Bagian Ketujuh)

154	8	500	94.50
155	8	500	94.50
156	8	500	94.75
157	8	500	94.75
158	8	500	94.00
159	8	500	94.25
160	8	500	94.50
161	10	500	94.25
162	10	500	94.00
163	10	500	94.50
164	10	500	94.50
165	10	500	94.50
166	10	500	94.50
167	10	500	94.00
168	10	500	93.33
169	10	500	93.33
170	10	500	94.00
171	12	500	94.50
172	12	500	94.25
173	12	500	94.50
174	12	500	93.75
175	12	500	93.75
176	12	500	94.00
177	12	500	94.25
178	12	500	94.25
179	12	500	94.25
180	12	500	94.00

Tabel A.8 Hasil Uji Coba JST-SOM (Bagian kedelapan)

182	14	500	94.00
183	14	500	94.00
184	14	500	94.25
185	14	500	94.25
186	14	500	94.25
187	14	500	94.25
188	14	500	94.25
189	14	500	94.00
190	14	500	94.00
191	16	500	94.00
192	16	500	94.00
193	16	500	94.00
194	16	500	94.66
195	16	500	94.00
196	16	500	94.00
197	16	500	94.00
198	16	500	94.00
199	16	500	94.75
200	16	500	94.75

Tabel A.9 Hasil Uji Coba JST-SOM Subset 2 (Bagian Pertama)

1	8	400	94.00
2	8	400	94.50
3	8	400	94.00
4	8	400	94.50
5	8	400	95.00
6	8	400	95.00
7	8	400	95.50
8	8	400	95.50
9	8	400	95.50
10	8	400	95.50
11	10	400	95.90
12	10	400	94.50
13	10	400	94.50
14	10	400	95.90
15	10	400	94.75
16	10	400	94.75
17	10	400	95.90
18	10	400	95.00
19	10	400	95.00
20	10	400	95.50
21	12	400	94.50
22	12	400	94.50
23	12	400	94.50
24	12	400	94.00
25	12	400	94.50
26	12	400	94.50
27	12	400	94.00

Tabel A.10 Hasil Uji Coba JST-SOM Subset 2 (Bagian Kedua)

28	12	400	94.50
29	12	400	95.80
30	12	400	95.80
31	14	400	95.25
32	14	400	95.00
33	14	400	95.00
34	14	400	95.50
35	14	400	95.50
36	14	400	95.50
37	14	400	95.50
38	14	400	95.75
39	14	400	95.75
40	14	400	95.25
41	16	400	95.30
42	16	400	95.30
43	16	400	95.30
45	16	400	95.30
46	16	400	95.30
47	16	400	94.75
48	16	400	95.30
49	16	400	95.00
50	16	400	94.00
51	8	450	96.00
52	8	450	96.30
53	8	450	96.00
54	8	450	95.75
55	8	450	95.75

Tabel A.11 Hasil Uji Coba JST-SOM Subset 2 (Bagian Ketiga)

56	8	450	96.30
57	8	450	96.30
58	8	450	96.30
59	8	450	96.30
60	8	450	96.30
61	10	450	95.25
62	10	450	95.00
63	10	450	95.80
64	10	450	95.80
65	10	450	95.25
66	10	450	95.80
67	10	450	95.80
68	10	450	95.25
69	10	450	95.00
70	10	450	95.75
71	12	450	94.00
72	12	450	94.50
73	12	450	94.00
74	12	450	94.50
75	12	450	93.75
76	12	450	94.50
77	12	450	94.00
78	12	450	94.25
79	12	450	94.25
80	12	450	94.50
81	14	450	95.80
82	14	450	95.25

Tabel A.12 Hasil Uji Coba JST-SOM Subset 2 (Bagian Keempat)

83	14	450	96.40
84	14	450	96.40
86	14	450	96.40
87	14	450	95.80
88	14	450	96.50
89	14	450	96.00
90	14	450	96.00
91	16	450	94.00
92	16	450	94.20
93	16	450	94.20
94	16	450	94.20
95	16	450	94.00
96	16	450	94.00
97	16	450	94.00
98	16	450	94.20
99	16	450	94.00
100	16	450	94.00
101	8	475	94.50
102	8	475	94.00
103	8	475	94.50
104	8	475	93.75
105	8	475	93.75
106	8	475	94.00
107	8	475	94.25
108	8	475	94.50
109	8	475	94.25
110	8	475	94.00

Tabel A.13 Hasil Uji Coba JST-SOM Subset 2 (Bagian Kelima)

111	10	475	94.50
112	10	475	94.50
113	10	475	94.75
114	10	475	94.66
115	10	475	94.00
116	10	475	94.33
117	10	475	94.33
118	10	475	94.00
119	10	475	94.00
120	10	475	94.66
121	12	475	94.00
122	12	475	94.33
123	12	475	94.00
124	12	475	94.50
125	12	475	94.50
126	12	475	94.50
127	12	475	94.00
128	12	475	94.33
129	12	475	94.00
130	12	475	94.50
131	14	475	94.25
132	14	475	94.25
133	14	475	94.33
134	14	475	94.25
135	14	475	94.00
136	14	475	94.00
137	14	475	94.00

Tabel A.14 Hasil Uji COba JST-SOM Subset 2 (Bagian Keenam)

137	14	475	94.00
138	14	475	94.00
139	14	475	94.00
140	14	475	94.00
141	16	475	94.50
142	16	475	94.00
143	16	475	93.66
144	16	475	94.50
145	16	475	94.50
146	16	475	94.50
147	16	475	94.00
148	16	475	94.00
149	16	475	94.50
150	16	475	93.33
151	8	500	94.00
152	8	500	94.25
153	8	500	94.75
154	8	500	95.60
155	8	500	94.00
156	8	500	94.00
157	8	500	94.50
158	8	500	93.33
159	8	500	94.00
160	8	500	94.25
161	10	500	94.75
162	10	500	94.50
163	10	500	94.00

Tabel A.15 Hasil Uji COba JST-SOM Subset 2 (Bagian Ketujuh)

164	10	500	94.00
165	10	500	94.00
166	10	500	94.00
167	10	500	94.00
168	10	500	94.50
169	10	500	94.00
170	10	500	93.66
171	12	500	94.50
172	12	500	94.50
173	12	500	94.50
174	12	500	94.00
175	12	500	94.00
176	12	500	94.50
177	12	500	93.33
178	12	500	94.00
179	12	500	94.25
180	12	500	94.75
181	14	500	95.60
182	14	500	94.00
183	14	500	94.00
184	14	500	94.50
185	14	500	93.33
186	14	500	94.00
187	14	500	94.25
188	14	500	94.75
189	14	500	94.50

Tabel A.16 Hasil Uji COba JST-SOM Subset 2 (Bagian Kedelapan)

190	14	500	94.25
191	16	500	94.75
192	16	500	95.60
193	16	500	94.00
194	16	500	94.00
195	16	500	94.50
196	16	500	93.33
197	16	500	94.00
198	16	500	94.25
199	16	500	94.75
200	16	500	94.50

Tabel A.17 Hasil Uji Coba JST-SOM Subset 3 (Bagian Pertama)

1	8	400	94.00
2	8	400	94.50
3	8	400	94.00
4	8	400	94.50
5	8	400	95.00
6	8	400	95.00
7	8	400	95.50
8	8	400	95.75
9	8	400	95.75
10	8	400	95.75
11	10	400	95.00
12	10	400	94.50
13	10	400	94.50
14	10	400	95.00
15	10	400	94.75
16	10	400	94.75
17	10	400	95.25
18	10	400	95.00
19	10	400	95.00
20	10	400	95.50
21	12	400	95.75
22	12	400	95.75
23	12	400	95.75
24	12	400	95.00
25	12	400	94.50
26	12	400	94.50
27	12	400	95.00

Tabel A.18 Hasil Uji Coba JST-SOM Subset 3 (Bagian Kedua)

28	12	400	95.50
29	12	400	96.00
30	12	400	96.00
31	14	400	95.25
32	14	400	95.00
33	14	400	95.00
34	14	400	95.50
35	14	400	95.50
36	14	400	95.50
37	14	400	95.50
38	14	400	95.75
39	14	400	95.75
40	14	400	95.25
41	16	400	95.25
42	16	400	95.00
43	16	400	95.00
45	16	400	95.25
46	16	400	95.25
47	16	400	94.75
48	16	400	95.25
49	16	400	95.00
50	16	400	94.00
51	8	450	94.75
52	8	450	94.75
53	8	450	94.00
54	8	450	94.75
55	8	450	94.75

Tabel A.19 Hasil Uji Coba JST-SOM Subset 3 (Bagian Ketiga)

56	8	450	94.75
57	8	450	94.50
58	8	450	94.75
59	8	450	94.00
60	8	450	94.25
61	10	450	94.25
62	10	450	94.00
63	10	450	94.50
64	10	450	94.50
65	10	450	94.25
66	10	450	94.50
67	10	450	94.50
68	10	450	94.25
69	10	450	94.00
70	10	450	94.75
71	12	450	94.00
72	12	450	94.50
73	12	450	94.00
74	12	450	94.50
75	12	450	93.75
76	12	450	94.50
77	12	450	94.00
78	12	450	94.25
79	12	450	94.25
80	12	450	94.50
81	14	450	93.50
82	14	450	93.25

Tabel A.20 Hasil Uji Coba JST-SOM Subset 3 (Bagian Keempat)

83	14	450	94.00
84	14	450	93.25
86	14	450	93.25
87	14	450	93.75
88	14	450	93.50
89	14	450	94.00
90	14	450	95.00
91	16	450	95.00
92	16	450	94.25
93	16	450	94.50
94	16	450	94.25
95	16	450	95.00
96	16	450	94.50
97	16	450	94.50
98	16	450	94.75
99	16	450	95.00
100	16	450	95.00
101	8	475	94.50
102	8	475	94.00
103	8	475	94.50
104	8	475	93.75
105	8	475	93.75
106	8	475	94.00
107	8	475	94.25
108	8	475	94.50
109	8	475	94.25
110	8	475	94.00

Tabel A.21 Hasil Uji Coba JST-SOM Subset 3 (Bagian Kelima)

111	10	475	94.50
112	10	475	94.50
113	10	475	94.75
114	10	475	94.66
115	10	475	94.00
116	10	475	94.33
117	10	475	94.33
118	10	475	94.00
119	10	475	94.00
120	10	475	94.66
121	12	475	94.00
122	12	475	94.33
123	12	475	94.00
124	12	475	94.50
125	12	475	94.50
126	12	475	94.50
127	12	475	94.00
128	12	475	94.33
129	12	475	94.00
130	12	475	94.50
131	14	475	94.25
132	14	475	94.25
133	14	475	94.33
134	14	475	94.25
135	14	475	94.00
136	14	475	94.00
137	14	475	94.00

Tabel A.22 Hasil Uji COba JST-SOM Subset 3 (Bagian Keenam)

137	14	475	94.00
138	14	475	94.00
139	14	475	94.00
140	14	475	94.00
141	16	475	94.50
142	16	475	94.00
143	16	475	93.66
144	16	475	94.50
145	16	475	94.50
146	16	475	94.50
147	16	475	94.00
148	16	475	94.00
149	16	475	94.50
150	16	475	93.33
151	8	500	94.00
152	8	500	94.25
153	8	500	94.75
154	8	500	95.60
155	8	500	94.00
156	8	500	94.00
157	8	500	94.50
158	8	500	93.33
159	8	500	94.00
160	8	500	94.25
161	10	500	94.75
162	10	500	94.50
163	10	500	94.00

Tabel A.23 Hasil Uji COba JST-SOM Subset 3 (Bagian Ketujuh)

164	10	500	94.00
165	10	500	94.00
166	10	500	94.00
167	10	500	94.00
168	10	500	94.50
169	10	500	94.00
170	10	500	93.66
171	12	500	94.50
172	12	500	94.50
173	12	500	94.50
174	12	500	94.00
175	12	500	94.00
176	12	500	94.50
177	12	500	93.33
178	12	500	94.00
179	12	500	94.25
180	12	500	94.75
181	14	500	95.60
182	14	500	94.00
183	14	500	94.00
184	14	500	94.50
185	14	500	93.33
186	14	500	94.00
187	14	500	94.25
188	14	500	94.75
189	14	500	94.50

Tabel A.24 Hasil Uji COba JST-SOM Subset 3 (Bagian Kedelapan)

190	14	500	94.25
191	16	500	94.75
192	16	500	95.60
193	16	500	94.00
194	16	500	94.00
195	16	500	94.50
196	16	500	93.33
197	16	500	94.00
198	16	500	94.25
199	16	500	94.75
200	16	500	94.50

Tabel A.25 Hasil Uji Coba JST-SOM Subset 4 (Bagian Pertama)

1	8	400	94.00
2	8	400	94.50
3	8	400	94.00
4	8	400	94.50
5	8	400	95.00
6	8	400	95.00
7	8	400	95.50
8	8	400	95.75
9	8	400	95.75
10	8	400	95.75
11	10	400	95.00
12	10	400	94.50
13	10	400	94.50
14	10	400	95.00
15	10	400	94.75
16	10	400	94.75
17	10	400	95.25
18	10	400	95.00
19	10	400	95.00
20	10	400	95.50
21	12	400	95.75
22	12	400	95.75
23	12	400	95.75
24	12	400	95.00
25	12	400	94.50
26	12	400	94.50
27	12	400	95.00

Tabel A.26 Hasil Uji Coba JST-SOM Subset 4 (Bagian Kedua)

28	12	400	95.50
29	12	400	96.00
30	12	400	96.00
31	14	400	95.25
32	14	400	95.00
33	14	400	95.00
34	14	400	95.50
35	14	400	95.50
36	14	400	95.50
37	14	400	95.50
38	14	400	95.75
39	14	400	95.75
40	14	400	95.25
41	16	400	95.25
42	16	400	95.00
43	16	400	95.00
45	16	400	95.25
46	16	400	95.25
47	16	400	94.75
48	16	400	95.25
49	16	400	95.00
50	16	400	94.00
51	8	450	94.75
52	8	450	94.75
53	8	450	94.00
54	8	450	94.75
55	8	450	94.75

Tabel A.27 Hasil Uji Coba JST-SOM Subset 4 (Bagian Ketiga)

56	8	450	94.75
57	8	450	94.50
58	8	450	94.75
59	8	450	94.00
60	8	450	94.25
61	10	450	94.25
62	10	450	94.00
63	10	450	94.50
64	10	450	94.50
65	10	450	94.25
66	10	450	94.50
67	10	450	94.50
68	10	450	94.25
69	10	450	94.00
70	10	450	94.75
71	12	450	94.00
72	12	450	94.50
73	12	450	94.00
74	12	450	94.50
75	12	450	93.75
76	12	450	94.50
77	12	450	94.00
78	12	450	94.25
79	12	450	94.25
80	12	450	94.50
81	14	450	93.50
82	14	450	93.25

Tabel A.28 Hasil Uji Coba JST-SOM Subset 4 (Bagian Keempat)

83	14	450	94.00
84	14	450	93.25
86	14	450	93.25
87	14	450	93.75
88	14	450	93.50
89	14	450	94.00
90	14	450	95.00
91	16	450	95.00
92	16	450	94.25
93	16	450	94.50
94	16	450	94.25
95	16	450	95.00
96	16	450	94.50
97	16	450	94.50
98	16	450	94.75
99	16	450	95.00
100	16	450	95.00
101	8	475	94.50
102	8	475	94.00
103	8	475	94.50
104	8	475	93.75
105	8	475	93.75
106	8	475	94.00
107	8	475	94.25
108	8	475	94.50
109	8	475	94.25
110	8	475	94.00

Tabel A.29 Hasil Uji Coba JST-SOM Subset 4 (Bagian Kelima)

111	10	475	94.50
112	10	475	94.50
113	10	475	94.75
114	10	475	94.66
115	10	475	94.00
116	10	475	94.33
117	10	475	94.33
118	10	475	94.00
119	10	475	94.00
120	10	475	94.66
121	12	475	94.00
122	12	475	94.33
123	12	475	94.00
124	12	475	94.50
125	12	475	94.50
126	12	475	94.50
127	12	475	94.00
128	12	475	94.33
129	12	475	94.00
130	12	475	94.50
131	14	475	94.25
132	14	475	94.25
133	14	475	94.33
134	14	475	94.25
135	14	475	94.00
136	14	475	94.00
137	14	475	94.00

Tabel A.30 Hasil Uji Coba JST-SOM Subset 4 (Bagian Keenam)

137	14	475	94.00
138	14	475	94.00
139	14	475	94.00
140	14	475	94.00
141	16	475	94.50
142	16	475	94.00
143	16	475	93.66
144	16	475	94.50
145	16	475	94.50
146	16	475	94.50
147	16	475	94.00
148	16	475	94.00
149	16	475	94.50
150	16	475	93.33
151	8	500	94.00
152	8	500	94.25
153	8	500	94.75
154	8	500	95.60
155	8	500	94.00
156	8	500	94.00
157	8	500	94.50
158	8	500	93.33
159	8	500	94.00
160	8	500	94.25
161	10	500	94.75
162	10	500	94.50
163	10	500	94.00

Tabel A.31 Hasil Uji Coba JST-SOM Subset 4 (Bagian Ketujuh)

164	10	500	94.00
165	10	500	94.00
166	10	500	94.00
167	10	500	94.00
168	10	500	94.50
169	10	500	94.00
170	10	500	93.66
171	12	500	94.50
172	12	500	94.50
173	12	500	94.50
174	12	500	94.00
175	12	500	94.00
176	12	500	94.50
177	12	500	93.33
178	12	500	94.00
179	12	500	94.25
180	12	500	94.75
181	14	500	95.60
182	14	500	94.00
183	14	500	94.00
184	14	500	94.50
185	14	500	93.33
186	14	500	94.00
187	14	500	94.25
188	14	500	94.75
189	14	500	94.50

Tabel A.32 Hasil Uji Coba JST-SOM Subset 4 (Bagian Kedelapan)

190	14	500	94.25
191	16	500	94.75
192	16	500	95.60
193	16	500	94.00
194	16	500	94.00
195	16	500	94.50
196	16	500	93.33
197	16	500	94.00
198	16	500	94.25
199	16	500	94.75
200	16	500	94.50

BIODATA PENULIS

Naufal Aulia Rizal, biasa dipanggil Naufal, lahir di Purwakarta pada tanggal 30 April 1992, merupakan anak pertama dari tiga bersaudara. Penulis telah menempuh pendidikan mulai dari SD AL-Falah Surabaya (1998-2004), SMP Al-Azhar 2 Jakarta (2004-2007), SMA Al-Azhar 2 Jakarta (2007-2010), dan pada tahun 2010 penulis meneruskan pendidikannya di Teknik Informatika ITS.

Penulis aktif di organisasi Himpunan Mahasiswa Teknik Computer - Informatika (HMTC) ITS. Penulis Semasa perkuliahan sering mengikuti perlombaan salah satunya adalah GEMASTIK V dengan menjadi finalis Data Mining.

Dalam perkuliahan, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV) dan tertarik pada hal yang berhubungan dengan *data mining* dan *machine learning*. Penulis dapat dihubungi melalui email di naufal.076@gmail.com