repository.ub.ac.id

IMPLEMENTASI METODE JARINGAN SARAF TIRUAN *BACKPROPAGATION* PADA PREDIKSI *PAYLOAD* 4G DI TELKOMSEL JEMBER

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

M Rikzal Humam Al Kholili

NIM : 165150209111011

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2018

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PENGESAHAN

IMPLEMENTASI METODE JARINGAN SARAF TIRUAN *BACKPROPAGATION*
PADA PREDIKSI *PAYLOAD* 4G DI TELKOMSEL

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
M Rikzal Humam Alkholili
NIM: 165150209111011

Skripsi ini telah diuji dan dinyatakan lulus pada
26 Desember 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Budi Darma Setiawan S.Kom., M.Cs.	Randy Cahya Wihandika, S.ST., M.Kom.
NIP: 19841015 201404 1 002	NIK: 2014058 80206 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PERNYATAAN ORISINALITAS

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

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

Malang, 1 Januari 2019

M Rikzal Humam Alkholili

NIM: 165150209111011

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat, karunia dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "IMPLEMENTASI METODE JARINGAN SARAF TIRUAN *BACKPROPAGATION* PADA PREDIKSI *PAYLOAD* 4G DI TELKOMSEL JEMBER". Skripsi ini diajukan sebagai ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer (FILKOM), Program Studi Teknik Informatika, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan rasa terimakasih kepada:

1. Budi Darma Setiawan, S.Kom., M.Cs. selaku dosen pembimbing pertama skripsi yang telah banyak meluangkan waktu dan juga memberikan banyak pengarahan bagi penulis.
2. Randy Cahya Wihandika, S.ST., M.Kom. selaku dosen pembimbing kedua skripsi yang telah banyak meluangkan waktu dan juga memberikan banyak pengarahan bagi penulis.
3. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
5. Orang tua yang telah memberikan semangat dan juga memberikan banyak bantuan moril maupun materiil selama perkuliahan hingga penyelesaian skripsi ini.
6. Segenap staff dan karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi.

Penulis menyadari bahwa dalam menyelesaikan skripsi ini, tentunya masih jauh dari kesempurnaan. Maka dari itu kritik dan saran yang membangun sangat diharapkan untuk kesempurnaan karya berikutnya.

Semoga penyusunan skripsi ini dapat berguna, bermanfaat bagi penulis maupun pembaca.

Malang, 28 November 2018

Penulis

M Rikzal Humam Alkholili

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRAK

PT. Telkomsel adalah sebuah perusahan pelayanan jasa telekomunikasi seluler terbesar di Indonesia yang memiliki lebih dari satu juta pelanggan yang tersebar di seluruh penjuru daerah Indonesia. Pelanggan Telkomsel tiap tahun mengalami kenaikan dimana kenaikan pelanggan ini akan berpengaruh terhadap *payload* yang dibagi tiap daerah. Jika *payload* pada daerah tertentu mengalami kelebihan pemakai maka akan terjadi *over lagging*. Untuk mengatasi masalah tersebut, peneliti menggunakan metode *backpropagation* yang akan digunakan untuk memprediksi jumlah *payload* 4G yang akan terjadi di hari berikutnya, sehingga pegawai Telkomsel dapat mengetahui daerah mana yang harus dilakukan *upgrade payload* 4G pada hari ke berapa. Penelitian ini menggunakan metode *backpropagation* dengan arsitektur jaringan 4 neuron *node* sebagai layer masukan, 6 neuron *node* sebagai layer tersembunyi dan 1 *node* sebagai layer keluaran. Setelah beberapa pengujian dilakukan, peneliti menemukan nilai yang paling optimal untuk masalah prediksi *payload* 4G di Telkomsel dengan nilai *learning rate* sebesar 0,1 dan iterasi maksimum sebanyak 1000 serta dengan menggunakan 40 data sebagai data latih dan 18 data sebagai data uji. Sehingga menghasilkan sebuha nilai MAPE (*Mean Absolute Percentage Error*) sebesar 6,0154830745999%. Dengan nilai *error* sebesar 6,0154830745999% dapat dikatakan bahwa nilai prediksi pada masalah *payload* 4G di Telkomsel cukup akurat, namun pada beberapa data yang lain bisa saja tidak memperoleh hasil yang akurat.

Kata Kunci: Implementasi, Jaringan Saraf Tiruan , *Payload* 4G, *Backpropagation*

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

PT. Telkomsel is the largest cellular telecommunications services company in Indonesia that has more than one million customers spread throughout the Indonesian region. Telkomsel customers each year experience an increase where the increase in customers will affect the payload divided by each region. If the payload in a certain area experiences excess user, there will be over lagging. To overcome this problem, researchers used the backpropagation method which will be used to predict the amount of 4G payload that will occur the next day, so that Telkomsel employees can find out which areas to upgrade the 4G payload to on what day. This study uses the backpropagation method with the network architecture of 4 nodes as the input layer, 6 nodes as hidden layers and 1 node as the output layer. After several tests were carried out, the researchers found the most optimal value for the 4G payload prediction problem in Telkomsel with a learning rate value of 0,1 and a maximum iteration of 1000 and by using 40 data as training data and 18 data as test data. So that it produces an MAPE value (Mean Absolute Percentage Error) of 6,0154830745999%. With an error value of 6,0154830745999% it can be said that the predictive value of the problem of 4G payload in Telkomsel is quite accurate, but in some other data it may not get accurate results.

Keyword : Implementation, Neural Network, Payload 4G, Backpropagation

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xi
DAFTAR KODE PROGRAM xii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 2
1.4 Manfaat 3
1.5 Batasan Masalah 3
1.6 Sistematika Pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Tinjauan Pustaka 5
2.2 4G LTE (*Long Term Evolution*) 7
2.3 *Payload* 8
2.4 Jaringan Saraf Tiruan 8
2.5 *Backpropagation* 11
BAB 3 METODOLOGI 16
3.1 Tipe Penelitian 16
3.2 Strategi Penelitian 16
3.3 Partisipan Penelitian 16
3.4 Lokasi Penelitian 16
3.5 Teknik Pengumpulan Data 16
3.6 Data Penelitian 17
3.7 Teknik Analisis Data 17

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

3.8 Implementasi Algoritme 17
BAB 4 PERANCANGAN 18
4.1 Deskripsi Masalah 18
4.2 Desain Algoritme 19
4.2.1 Normalisasi Data 20
4.2.2 *Backpropagation* 21
4.2.3 Denormalisasi Data 22
4.3 Perhitungan Manual Algoritme *Backpropagation* 23
4.4 Perancangan Antarmuka 30
4.4.1 Perancangan Halaman *Home* 31
4.4.2 Perancangan Halaman Latih 32
4.4.3 Perancangan Halaman Uji 33
4.4.4 Perancangan Halaman Ramal 34
BAB 5 IMPLEMENTASI 35
5.1 Implementasi Program Metode *Backpropagation* 35
5.1.1 Proses *Load* Data 35
5.1.2 Proses Normalisasi Data 36
5.1.3 Proses *Extract* Data 37
5.1.4 Set Bobot 39
5.1.5 Proses Pelatihan 40
5.1.6 Proses Pengujian 43
5.2 Implementasi Tampilan Antarmuka 45
5.2.1 Implementasi Halaman *Home* 45
5.2.2 Implementasi Halaman Latih 46
5.2.3 Implementasi Halaman Uji 46
5.2.4 Implementasi Halaman Ramal 47
BAB 6 ANALISIS DAN PENGUJIAN 49
6.1 Analisis dan Pengujian Jumlah Data Terhadap Nilai MSE 49
6.2 Analisis dan Pengujian *Learning Rate* Terhadap Nilai MSE 50
6.3 Analisis dan Pengujian Jumlah Iterasi Terhadap Nilai MSE 51
6.4 Analisis dan Pengujian *K-Fold Validation* Terhadap Nilai MSE 52
6.5 Analisis dan Pengujian Hasil Prediksi 53

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 7 PENUTUP 55
7.1 Kesimpulan 55
7.2 Saran 55
DAFTAR PUSTAKA 56

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

Tabel 4.1 Data Jumlah *Payload* 4G 18
Tabel 4.2 Inisialisasi Bobot *v* 23
Tabel 4.3 Inisialisasi Bobot *w* 23
Tabel 4.4 Data Normalisasi 23
Tabel 4.5 Hasil Nilai *Z_netj* 24
Tabel 4.6 Nilai *Zj* 24
Tabel 4.7 Nilai Δwkj 25
Tabel 4.8 Nilai δ_netj 26
Tabel 4.9 Nilai δj 26
Tabel 4.10 Nilai Δvji 27
Tabel 4.11 Nilai $wkjbaru$ 27
Tabel 4.12 Nilai $vjibaru$ 28
Tabel 4.13 Bobot Akhir wkj 28
Tabel 4.14 Bobot Akhir vji 28
Tabel 4.15 Hasil Nilai *y* 29
Tabel 4.16 Hasil Denormalisasi Nilai *Y* 29
Tabel 4.18 Hasil Aktual dengan Hasil Prediksi 30
Tabel 4.19 Hasil *Error* Dengan MSE 30
Tabel 6.1 Pengujian Jumlah Data Terhadap Nilai MSE 49
Tabel 6.2. Pengujian *Learning Rate* Terhadap Nilai MSE 50
Tabel 6.3. Pengujian Jumlah Iterasi Terhadap Nilai MSE 51
Tabel 6.4 Pengujian *K-Fold Validation* Terhadap Nilai MSE 53

DAFTAR GAMBAR

Gambar 2.1 Arsitektur Jaringan Dengan Lapisan Tunggal 10
Gambar 2.2 Arsitektur Jaringan Saraf Dengan Lapisan Banyak 10
Gambar 2.4 Arsitektur Jaringan Dengan Lapisan Kompetitif 11
Gambar 4.1 *Flowchart* Desain Algoritme 19
Gambar 4.2 *Flowchart* Normalisasi Data 20
Gambar 4.3 *Flowchart* Algoritme *Backpropagation* 21
Gambar 4.4 Proses Denormalisasi Data 22
Gambar 4.5 Perancangan Halaman *Home* 31
Gambar 4.6 Perancangan Halaman Latih 32
Gambar 4.7 Perancangan Halaman Uji 33
Gambar 4.8 Perancangan Halaman Ramal 34
Gambar 5.1 Tampilan Halaman *Home* 45
Gambar 5.2 Tampilan Halaman Latih 46
Gambar 5.3 Tampilan Halaman Uji 47
Gambar 5.4 Tampilan Halaman Ramal 47
Gambar 6.1 Grafik Pengujian Jumlah Data 50
Gambar 6.2 Pengujian *Learning Rate* 51
Gambar 6.3 Pengujian Iterasi Terhadap Nilai MSE Latih 52
Gambar 6.4 Pengujian *K-Fold Validation* Terhadap Nilai MSE 53
Gambar 6.5 Pengujian Hasil 54

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR KODE PROGRAM

Kode Program 5.1 *Load* Data 35
Kode Program 5.2 Normalisasi Data 36
Kode Program 5.3 *Extract* Data 37
Kode Program 5.4 Inisialisasi Bobot 39
Kode Program 5.5 Proses Pelatihan 40
Kode Program 5.6 Proses Pengujian 43

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

BAB 1 PENDAHULUAN

1.1 Latar Belakang

PT. Telekomunikasi Seluler Indonesia merupakan *provider* telekomunikasi seluler yang paling besar dan paling banyak menyebar di seluruh kawasan Indonesia hingga daerah yang terpencil sekalipun, hal ini dapat dibuktikan pada halaman resmi telkomsel yaitu www.telkomsel.com (2017) yang menyatakan bahwa banyaknya pelanggan Telkomsel yang mencapai 178 juta pelanggan serta 146 ribu BTS yang tersebar di seluruh pelosok Indonesia.

Sudah sepatutnya PT. Telekomunikasi Seluler memiliki jumlah pelanggan yang semakin bertambah, hal ini dapat dibuktikan pada situs https://id.techinasia.com (2014) yang menyatakan bahwa pada kuartal 1 tahun 2014 memiliki pengguna sebanyak 132,7 juta dan meningkat pada kuartal 3 tahun 2014 yakni sebanyak 139,2 juta pengguna. Pertambahan pelanggan tersebut dapat mempengaruhi pertambahan penggunaan *payload* 4G yang mengakibatkan gangguan jika tidak dilakukan *upgrade* sistem dengan segera. Sudah sepatutnya bagi pihak Telkomsel dapat memprediksi jumlah penggunaan *payload* 4G, sehingga pihak Telkomsel dapat merencanakan dan mengetahui site mana yang perlu dilakukan *upgrade* sistem di kemudian hari.

Prediksi adalah suatu cara untuk memperkirakan suatu hal yang akan terjadi di masa yang akan datang (Taylor, 2003). Untuk dapat memprediksi suatu keadaan yang akan terjadi, terdapat beberapa metode yang dapat melakukan prediksi dengan menggunakan perhitungan matematis berdasarkan rumus yang ada. Berbagai metode tersebut ialah metode *fuzzy time series* yang digunakan oleh Elfajar (2017) untuk kasus peramalan jumlah kunjungan wisatawan di Kota Batu. Peneliti menggunakan data pengunjung perbulan yang diperoleh dari Dinas Pariwisata Kota Batu. Berdasarkan hasil pengujian yang sudah dilakukan oleh peneliti dengan menggunakan *Average Forecasting Error Rate* (AFER) menghasilkan nilai eror rata–rata sebesar 0.0056% dengan menggunakan data latih 60. Metode lain yang dapat digunakan untuk meramalkan suatu permasalahan adalah metode *support vector regression* oleh Raharyani (2018) untuk kasus prediksi jumlah pengunjung pariwisata. Data yang digunakan adalah jumlah data pengunjung pariwisata bulanan sebanyak 72 data dari tahun 2010 sampai 2015. Berdasarkan pengujian yang dilakukan oleh peniliti menghasilkan nilai MAPE terbaik yaitu 6,98% dengan parameter *sigma* = 925,8409 *lambda* = 0,3868, *cLR* = 0,0802, *epsilon* = 1,27E-10, *complexity* = 3234,539, jumlah iterasi maksimal 5000.

Metode jaringan saraf tiruan backpropagation merupakan salah satu metode yang dapat digunakan untuk memprediksi suatu kondisi yang ada. Ada beberapa penelitian yang menggunakan metode jaringan saraf tiruan untuk peramalan suatu kondisi. Rachman (2018) melakukan penelitian tentang

UNIVERSITAS BRAWIJAYA

peramalan produksi gula dengan menggunakan metode jaringan saraf tiruan backpropagation dengan menggunakan 4 *input layer,* 3 *hidden layer* serta 1 *output layer* dan menghasilkan nilai MAPE terendah sebesar 17,85% dengan 800 iterasi pada pengujian jumlah iterasi maksimum serta nilai MAPE terendah sebesar 17,38 % dengan nilai *learning rate* 0,4 pada pengujian *learning rate*. Penelitian yang serupa dilakukan oleh Susanti (2014) yang membahas tentang penerapan model *neural network backpropagation* untuk prediksi harga ayam dengan data bertipe *time series* dari bulan Agustus 2010 sampai bulan Mei 2013. Peneliti menggunakan arsitektur jaringan 4 *input layer*, *10 hidden layer* dan 1 *output layer* yang menghasilkan nilai MSE sebesar 0,0113 dengan menggunakan *learning rate* 0,05 dan maksimal iterasi sebanyak 5000.

Sehingga dengan melakukan penelitian yang berjudul "Implementasi Metode Jaringan Saraf Tiruan Pada Peramalan *Payload* 4G Di Telkomsel Jember" dapat membantu pihak Telkomsel dalam hal menentukan jumlah *payload* 4G di kemudian hari.

1.2 Rumusan Masalah

Berdasarkan Latar Belakang yang ada, maka dapat disusun Rumusan Masalah sebagai berikut:

1. Bagaimana karakteristik dari variabel algoritme *backpropagation* agar mendapatkan hasil prediksi yang paling optimal pada kasus prediksi *payload 4G* di Telkomsel?
2. Bagaimana tingkat akurasi Metode Jaringan Saraf Tiruan Backpropagation untuk peramalan *payload* 4G di Telkomsel Jember?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Untuk mengetahui karakteristik dari variabel algoritme *backpropagation* agar mendapatkan hasil prediksi yang paling optimal pada kasus prediksi *payload* 4G di Telkomsel.
2. Untuk mengetahui tingkat akurasi Metode Jaringan Saraf Tiruan Backpropagation untuk peramalan *payload* 4G di Telkomsel Jember

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Untuk pihak Telkomsel dapat mengetahui *site* mana yang perlu di-*upgrade* di kemudian hari.
2. Untuk pihak peneliti dapat mengetahui cara menerapkan metode Jaringan Saraf Tiruan *Backpropagation* pada kasus tertentu dan juga dapat mengetahui tingkat akurasi dari Metode Jaringan Saraf Tiruan *Backpropagation*.

1.5 Batasan Masalah

Agar penelitian dapat terarah maka perlu dibuat sebuah batasan masalah. Adapun batasan masalah dari penelitian ini adalah sebagai berikut :

1. Data diperoleh dari pihak Telkomsel langsung berupa data *payload* dari bulan Desember 2017 sampai Januari 2018 dan berupa data per hari.
2. Metode yang digunakan adalah metode Jaringan Saraf Tiruan *Backpropagation*.

1.6 Sistematika Pembahasan

Pada subbab sistematika penulisan diterangkan gambaran umum dari penulisan yang dilakukan pada penelitian ini. Adapun kerangka dari sistematika penulisan pada penelitian adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab 1 menerangkan tentang deskripsi masalah yang ada dan solusi dari masalah tersebut yang ada pada subbab latar belakang. Selain masalah dan solusi pada bab 1 ini menerangkan tentang metode yang digunakan untuk melakukan penelitian ini serta tujuan dan manfaat dari peneliti melakukan penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Pada bab 2 berisi tentang pembahasan beberapa penelitian yang serupa dengan penelitian yang dilakukan sebelumnya untuk memperkuat hipotesa dalam melakukan penelitian. Selain dari itu pada bab 2 ini, berisi dasar teori yang mendasari teori–teori yang digunakan dalam penelitian ini.

BAB III METODOLOGI PENELITIAN

Pada bab metodologi penelitian berisi penjelasan tentang kerangka berfikir dalam melakukan penelitian ini, dimulai dari proses studi literatur sampai penarikan kesimpulan.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB IV ANALISIS DAN PERANCANGAN

Bab perancangan merupakan suatu bab yang menjelaskan tentang pengembangan secara teknis dari apa yang akan diterapkan pada komputer atau aplikasi *desktop*. Pada bab perancangan juga berisi tentang perhitungan manual dari metode yang akan digunakan.

BAB V IMPLEMENTASI

Pada bab implementasi berisi penjelasan tentang aplikasi yang sudah dibuat dengan menerapkan metode yang digunakan seperti *source code,* tampilan antarmuka dan tata cara menggunakan aplikasi yang sudah dibuat.

BAB VI PENGUJIAN

Pada bab pengujian berisi tentang pengujian terhadap metode yang sudah diterapkan pada kasus yang di ambil. Apakah metode yang digunakan cocok untuk kasus yang di ambil atau tidak dan bagaimana tingkat akurasi metode tersebut.

BAB VII PENUTUP

Bab 7 adalah bab penutup yakni bab yang berisi tentang penarikan kesimpulan terkait dengan penelitian yang sudah dilakukan, dan juga berisi saran untuk peneliti yang lain jika ada peniliti yang ingin mengembangkan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Penelitian yang pertama dilakukan oleh Rachman (2018) yang menggunakan metode Jaringan Saraf Tiruan Backpropagation untuk meramalkan produksi gula di PG Candi Baru Sidoarjo. Pada penelitian ini peneliti menggunakan arsitektur jaringan berupa 4 *input layer,* 3 *hidden layer* dan 1 *output layer*. Juga menghasilkan sebesar 17,85 % nilai MAPE terendah dengan maksimal 800 iterasi. Serta menghasilkan nilai *learning rate* sebesar 0,4 dengan nilai MAPE terendah yakni 17,38 %. Dengan menggunakan nilai *learning rate* 0,4 dan jumlah iterasi maksimal 800 maka menghasilkan MAPE 16,98%.

Penelitian selanjutnya dilakukan oleh Diputra (2018) yang menggunakan metode jaringan saraf tiruan untuk memprediksi tingkat keuntungan usaha peternakan itik albio petelur. Peneliti menggunakan nilai *learning rate* 0,8 dengan jumlah *neuron* pada *hidden layer* sebanyak 17 serta menggunakan data latih sebanyak 90% dari total data latih yang ada. Dan menghasilkan nilai MAPE terbaik yaitu 25,7852%.

Penelitian yang ketiga dilakukan oleh Rindhani (2017) menggunakan metode *backpropagation* untuk meramal dosis pupuk berdasarkan karakteristik dan lingkungan tanaman jeruk siam. Peneliti menggunakan arsitektur jaringan dengan 3 *neuron input,* 5 *hidden layer* dan 3 *output layer* dengan nilai *learning rate* 0,3. Jumlah maksimal iterasi sebanyak 500 dengan data latih 56 dan data uji sebanyak 8. Menghasilkan nilai *Mean Absolute Precentage Error* (MAPE) sebesar 9,178% .

Penelitian berikutnya dilakukan oleh Yohanes (2015) yang menggunakan metode *backpropagation neural network* untuk menentukan upah minimum Kota berdasarkan tingkat inflasi. Peneliti menggunakan metode *Backpropagation Neural Network* untuk meramal nilai UMK pada tahun berikutnya. Setelah melakukan pengujian, peneliti mendapatkan nilai yang paling optimal yaitu, jumlah iterasi sebanyak 80, jumlah *hidden layer* sebanyak 1, serta nilai *learning rate* sebesar 0,8 dengan menghasilkan nilai MSE = 0,07280534710552478.

Yang terakhir penelitan dilakukan oleh (2018) yang menggunakan metode *backpropagation* untuk meramal produksi kelapa sawit. Penelitian berlangsung di PT. Sandabi Indah Lestari yang merupakan salah satu perkebunan besar di Indonesia. Peneliti menggunakan metode *backpropagation* untuk memprediksi hasil produksi di tahun berikutnya. Jumlah data latih yang digunakan sebanyak 260 dengan data uji sebanyak 12, sedangkan nilai *learning rate* sebesar 0,4, jumlah *hidden layer* sebanyak 5, jumlah iterasi maksimum sebanyak 900 iterasi. Sehingga menghasilkan nilai MAPE sebesar 10,0047%.

No	Judul	Objek	Metode	Hasil
1	Peramalan Produksi Gula Menggunakan Metode Jaringan Saraf Tiruan *Backpropagation* Pada PG Candi Baru Sidoarjo	Gula di PG Candi Baru Sidoajo	Jaringan Saraf Tiruan *Backpropagation*	Dengan menggunakan iterasi maksimum 800, *learning rate* 0,4 menghasilkan nilai Mape 16,98%
2	Prediksi Tingkat Keuntungan Usaha Peternakan Itik Alabio Petelur menggunakan Jaringan Saraf Tiruan *Backpropagation* (Kasus di Kabupaten Hulu Sungai Utara Kalimantan Selatan)	Itik alabio petelur di Kabupaten Hulu Sungai Utara Kalimantan Selatan	Jaringan Saraf Tiruan *Backpropagation*	Dengan menggunakan nilai *learning rate* sebesar 0,8, jumlah data latih sebesanyak 90% menghasilkan nilai MAPE 25,7852%
3	Peramalan Dosis Pupuk Berdasarkan Karakteristik dan Lingkungan Tanaman Jeruk Siam Menggunakan Metode	Dosis pupuk untuk tanaman jeruk siam	*Backpropagation*	Menggunakan nilai *learning rate* sebesar 0,3, iterasi sebanyak 500, 56 data latih serta 8 data uji menghasilkan nilai MAPE 9,178%

	Backpropagation			
4	Penentuan Upah Minimum Kota Berdasarkan Tingkat Inflasi Menggunakan *Backpropagation Neural Network* (BPNN)	Upah minimum berdasarkan tingkat inflasi	*Backpropagation Neural Network* (BPNN)	Menggunakan nilai *learning rate* sebesar 0,8 menghasilkan nilai MSE 0,07280534710552478.
5	Peramalan Produksi Kelapa Sawit Menggunakan Jaringan Saraf Tiruan Dengan Metode *Backpropagation*	Kelapa sawit	Jaringan Saraf Tiruan *Backpropagation*	Dengan menggunakan 260 data latih, 12 data uji, dan nilai *learning rate* sebesar 0,4 serta iterasi maksimum sebanyak 900 menghasilkan nilai MAPE sebesar 10,0047%

2.2 4G LTE (*Long Term Evolution*)

Long Term Evolution (LTE) adalah sebuah teknologi perkembangan dari 3G (*Third Generation*) yang memiliki transfer data *rate* yang lebih besar dari generasi yang sebelumnya. Teknologi LTE masih belum sepenuhnya bisa disebut teknologi 4 *Generation* tepatnya masih 3,9G. Kelebihan dari LTE diantaranya (Ariyanti, 2013):

a. *Latency*/*delay* lebih rendah
b. Data *rate* lebih tinggi
c. Mengingkatkan kapasitas dan *coverage*
d. *Cost-reduction*

WCMA adalah teknologi yang dikembangkan sebelum adanya teknologi LTE pada tahun 2003/2004 yang memiliki data *rate* sekitar 384 kbps *downlink*, 128 kbps *uplink* dan *round tip time* 150 ms. Setelah WCMA dibangun, pada tahun 2005/2006 muncul teknologi HSDPA (*High Speed Downlink Packet Access*) yang memiliki data *rate downlink* mencapai 14 Mbps, *uplink* hingga 5.7 Mbps. Kemudian pada tahun 2008/2009 3GPP me-*release* teknologi HSPA+ dengan data

rate mencapai 28Mbps untuk *downlink* dan *uplink* sebesar 11 Mbps. Teknologi yang terakhir yakni LTE yang di-*release* pada tahun 2010 yang memiliki data *rate uplink* sebesar 150Mbps. Teknologi LTE menguntungkan bagi perangkat *mobile* dan *home user* karena memiliki transmisi yang berkecepatan 100 Mbps, dan juga mungkin *home user* akan menggunakan koneksi telepon untuk *alternating* kedua karena jangkauan LTE lebih jauh (Ariyanti, 2013).

2.3 *Payload*

Ketika data dikirim melalui internet, setiap unit yang ditransmisikan mencakup informasi *header* dan data aktual yang dikirim. *Header* mengidentifikasikan sumber dan tujuan paket, sedangkan data sebenarnya disebut *payload*. Karena informasi *header*, atau data *overhead*, hanya digunakan dalam proses transmisi, kemudian dilepaskan dari paket saat mencapai tujuannya. Oleh karena itu *payload* adalah satu-satunya data yang diterima oleh sistem tujuan. Informasi pemberian sinyal dan kontrol digunakan untuk memantau, mengawasi, memelihara, dan mengatur dan mengendalikan jaringan dan komponennya. Data di atas digunakan untuk tujuan seperti penanganan, manajemen kemacetan, kontrol kesalahan, indikasi prioritas, dan panggilan *routing*, blok, *frame*, paket, dan sel yang berisi muatan.

Data–data yang perlu atau yang penting dibawa dengan paket atau transmisi yang lain. *Payload* tidak termasuk data "*overhead*" yang dibutuhkan untuk mendapatkan paket ke tujuannya. Untuk lapisan komunikasi yang membutuhkan beberapa data *overhead* untuk melakukan tugasnya, *payload* kadang-kadang dianggap mencakup bagian data *overhead* yang ditangani lapisan ini. Namun, dalam penggunaan yang lebih umum, payload adalah bit yang dikirimkan ke pengguna akhir di tempat tujuan.

2.4 Jaringan Saraf Tiruan

Jaringan saraf tiruan (JST) adalah suatu metode dalam ruang lingkup *Artificial Intelegence* (AI) yang memiliki arsitektur kerja menyerupai dengan otak manusia sehingga dapat menyelesaikan masalah yang kompleks dan memiliki pola–pola yang sulit dipahami. Pembangunan arsitektur jaringan saraf tiruan meniru prinsip kerja dari saraf biologi manusia yang dikembangkan menggunakan model matematis yang digeneralisasikan berdasarkan saraf dan kesadaran manusia (Amina & Irawan, 2013).

Jaringan saraf tiruan terbentuk dari jaringan saraf biologis manusia yang diringkas dalam bentuk rumus matematika dengan asumsi bahwa (Riadi, 2016):

1. Terdapat banyak neuron yang digunakan untuk memproses banyak informasi.
2. Sinyal–sinyal yang dikirimkan oleh neuron ke neuron yang lain dikirimkan melalui banyak penghubung.
3. Setiap penghubung memiliki nilai bobot yang akan berpengaruh terhadap kekuatan dari sinyal–sinyal.
4. Nilai *output* yang dihasilkan merupakan nilai perhitungan dari nilai *input* yang dihitung dengan menggunakan fungsi aktivasi dan selanjutnya akan dibandingkan dengan suatu batas ambang.

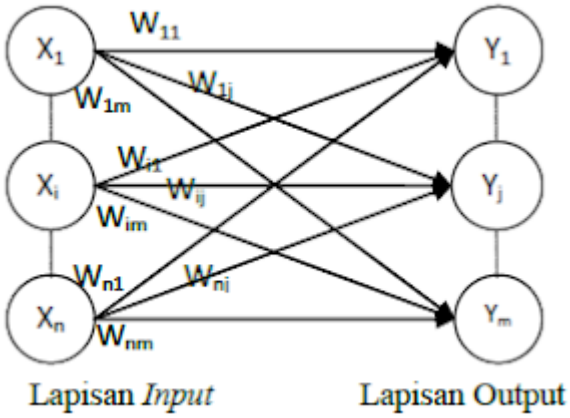
Pada jaringan saraf tiruan terdapat lapisan neuron yang merupakan kumpulan dari beberapa neuron–neuron yang ada. Setiap neuron memiliki informasi yang berbeda–beda. Informasi tersebut akan disampaikan pada setiap lapisan neuron selanjutnya. Lapisan–lapisan tersebut, yaitu:

1. *Input layer* adalah *node* yang berfungsi untuk sebuah masukan informasi dan kemudian menyebarkan informasi ke lapisan selanjutnya.
2. *Hidden layer* adalah *node* yang menerima informasi dari *input layer* dan kemudian diolah dengan fungsi transformasi *neuron*. *Hidden layer* terletak di antara *input layer* dan *output layer*.
3. *Output layer* adalah *node* yang menerima informasi dari *hidden layer* dan kemudian mengirimkan informasi tersebut kembali ke *hidden layer* yang merupakan hasil dari komputasi jaringan saraf tiruan.

Informasi yang terdapat pada *input layer* akan diteruskan pada *layer* selanjutnya yaitu *hidden layer*. Untuk menyampaikan informasi kepada *hidden layer* dibutuhkan sebuah penghubung antara *input layer* dengan *hidden layer*, penghubung tersebut berupa bobot–bobot yang nantinya akan mempengarui nilai dari informasi yang akan diteruskan kepada *hiddep layer.* Setelah melalui penghubung berupa bobot, maka akan dihitung dengan menggunakan fungsi aktivasi yang bertujuan untuk membatasi jarak *amplitude* yang dihasilkan oleh neuron agar menjadi nilai yang terbatas. Kemudian informasi akan diteruskan kepada *output layer* yang merupakan *layer* terahkhir.

Pada jaringan saraf tiruan tidak selalu memiliki tiga lapisan, namun ada juga yang memiliki dua lapisan. Menurut Haykin (2009), terdapat tiga jenis arsitektur jaringan saraf tiruan, yaitu:

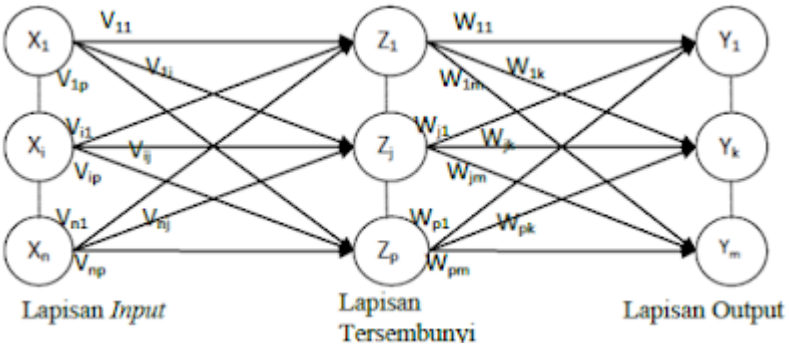
1. Jaringan dengan lapisan tunggal (*single layer perceptron*)



Gambar 2.1 Arsitektur Jaringan Dengan Lapisan Tunggal

Pada jaringan saraf tiruan dengan lapisan neuron tunggal, semua neuron–neuron yang ada dikumpulkan dalam sebuah satu lapisan yang paling sederhana dari semua arsitektur jaringan saraf tiruan yang ada. Arsitektur jaringan saraf tiruan dengan lapisan tunggal hanya memiliki satu lapisan masukan dan satu lapisan keluaran dimana sumber informasi yang berasal dari lapisan masukan dapat diestimasikan ke lapisan keluaran dan tidak bisa sebaliknya.

2. Jaringan dengan lapisan banyak (*multi layer perceptron*)

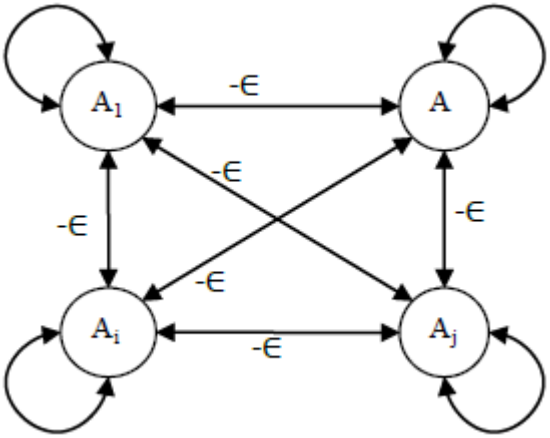


Gambar 2.2 Arsitektur Jaringan Saraf Dengan Lapisan Banyak

Jaringan saraf tiruan dengan banyak lapisan (*multi layer perceptron*) adalah sebuah arsitektur jaringan saraf tiruan yang memiliki satu atau lebih lapisan tersembunyi (*hidden layer*), satu lapisan masukan (*input layer*) dan satu lapisan keluaran (*output layer*). Arsitektur jaringan saraf tiruan dengan menggunakan satu atau lebih lapisan tersembunyi ini dapat memiliki kemampuan yang lebih baik

dengan arsitektur jaringan saraf tiruan tunggal yang tidak memiliki lapisan tersembunyi namun arsitektur jaringan saraf tiruan dengan lapisan banyak memiliki proses pelatihan yang lebih rumit. Pada beberapa kasus, jaringan saraf tiruan dengan lapisan lebih dari satu lapisan tersembunyi memiliki pelatihan yang lebih baik dibandingkan dengan arsitektur yang tidak memiliki lapisan tersembunyi karena pada arsitektur jaringan saraf tiruan dengan satu atau lebih lapisan tersembunyi memungkinkan dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh arsitektur yang tidak memiliki lapisan tersembunyi.

3. Jaringan dengan lapisan kompetitif (*competitive layer perceptron*)



Gambar 2.4 Arsitektur Jaringan Dengan Lapisan Kompetitif

Tipe arsitektur jaringan saraf tiruan dengan lapisan kompetitif merupakan tipe arsitektur jaringan saraf tiruan yang paling besar, karena pada arsitektur jaringan saraf tiruan dengan lapisan kompetitif setiap neuron–neuron yang berkumpul saling berlomba untuk menjadi sebuah neuron yang mendapatkan hak untuk aktif dan dipakai untuk proses pelatihan. Hubungan antar neuron yang ada pada arsitektur jaringan dengan lapisan kompetitif ini tidak ditunjukkan pada arsitektur jaringan saraf tiruan yang lain.

2.5 *Backpropagation*

Backpropagation adalah salah satu metode dari jaringan saraf tiruan yang dapat digunakan untuk masalah prediksi. Pada proses pelatihan dalam metode *backpropagation*, bobot yang digunakan untuk penghubung antar lapisan akan selalu dilakukan *update* agar mendapatkan bobot yang sesuai untuk permasalahan yang dipakai.

Pada proses metode *backpropagation* terdapat tiga fase (Siang, 2005), yaitu:

a. Fase 1 *Feedforward* (Propagasi maju)

Pada fase propagasi maju, akan dilakukan perhitungan dari layar masukan menuju layar tersembunyi dan dilanjutkan ke layar keluaran dengan menggunakan fungsi aktivasi yang sudah ditentukan.

b. Fase 2 *Backpropagation* (Propagasi mundur)

Fase propagasi mundur merupakan fase perhitungan *error* yang dihasilkan dari perhitungan propagasi maju. Nilai keluaran yang dihasilkan oleh propagasi maju akan dibandingkan dengan nilai target yang sudah ditentukan, kemudian semua yang berhubungan dengan layar keluaran akan dilakukan perhitungan *error*.

c. Fase 3 *Weight Update*(Perubahan bobot)

Pada fase perubahan bobot, bobot yang dihasilkan secara acak pada saat inisialisasi bobot, akan dilakukan perubahan pada fase tiga berdasarkan dengan nilai *error* yang dihasilkan pada saat propagasi mundur.

Proses perhitungan yang melewati propagasi maju, propagasi mundur dan perubahan bobot akan dilakukan secara berulang–ulang sampai nilai *error* sudah mencapai nilai yang sudah ditentukan atau nilai iterasi sudah mencapai pada jumlah iterasi yang ditentukan.

1. Langkah 0 : Melakukan inisialisasi bobot secara acak dengan rentang nilai antara 0 sampai dengan 1. Menentukan nilai *learning rate* dan iterasi.
2. Langkah 1 : Melakukan langkah 2 – 8 jika belum mencapai *stop condition*
3. Langkah 2 : Melakukan langkah 3 – 8 untuk perhitungan setiap data latih

Fase 1 : *Feedforward*

4. Langkah 3 : Setiap *input* x_i menerima sinyal dan meneruskan ke setiap unit di layar selanjutnya
5. Langkah 4 : Menghitung nilai z_net_j disetiap layar tersembunyi dengan menggunakan Persamaan (2.1)

$$z_net_j = v_{0j} + \sum_{i=0}^{n} x_i . v_{ij} \quad (2.1)$$

Dengan menggunakan fungsi sigmoid yang ada pada Persamaan (2.2) akan didapatkan nilai z_j

$$z_j = f\,(\,z_net_j\,) = \frac{1}{1+exp^{(-z_net_j)}} \quad (2.2)$$

Keterangan:
z_net_j = hasil perhitungan antara *input* x_i dengan bobot v_{ij}
v_{0j} = bobot bias layar masukan dengan layar tersembunyi

x_i = sinyal masukan ke–I (i=1,2….)
v_{ij} = bobot antara sinyal x_i dengan z_net_j
z_j = hasil dari z_net_j yang sudah dilakukan aktivasi
i = Indeks pada unit layar masukan
j = Indeks pada unit layar tersembunyi

6. Langhkah 5 : Menghitung setiap nilai y_net_k yang ada pada layar *output* (k = 1, 2 . . . n)

$$y_net_k = w_{0k} + \sum_{j=1}^{p} z_j \, . \, w_{jk} \quad (2.3)$$

Agar mendapatkan nilai y_k maka digunakan fungsi aktivasi *sigmoid* yang ditunjukkan pada Persamaan 2.4

$$y_k = f\,(\,y_net_k\,) = \frac{1}{1+exp^{(-y_net_k)}} \quad (2.4)$$

Keterangan:

y_net_k = Hasil perhitungan antara nilai *hidden layer* dengan bobot w_{jk}

w_{0k} = Bobot bias pada *hidden layer*

z_j = Nilai node *hidden layer* yang sudah diaktivasi

w_{jk} = Bobot antara *hidden layer* dengan *output layer*

y_k = Nilai *output layer* yang sudah diaktivasi

k = Jumlah *node* di *output layer*

Fase 2 : ***Backpropagation***

7. Langkah 6 : Menghitung nilai faktor δ pada setiap indeks k berdasarkan *error* pada setiap *output layer*

$$\delta_k = (t_k - y_k) f'(y_{net\,k}) = (t_k - y_k) y_k (1 - y_k) \quad (2.5)$$

Keterangan:

δ_k = Faktor perbaikan nilai bobot yang terhubung dengan *output layer*

y_net_k : Nilai *output layer* sebelum diaktivasi

t_k : Nilai target prediksi

y_k : Nilai yang dihasilkan dari perhitungan *feedforward*

k : Indeks *output layer*

δ_k digunakan untuk menghitung nilai perubahan bobot w_{jk} yang digunakan untuk meng-*update* bobot w_{jk} nanti.

Menghitung nilai perubahan bobot dengan menggunakan percepatan *alpha* (*learning rate*)

$$\Delta w_{jk} = \alpha\ \delta_k z_j \quad (2.6)$$

Keterangan:

Δw_{jk} = Nilai perubahan bobot w_{jk}

α = Nilai *learning rate*

δ_k = Faktor perbaikan bobot pada *output layer*

z_j = Nilai *output layer* yang sudah diaktivasi

8. Langkah 7 : Menghitung nilai δnet_j pada setiap indeks j berdasarkan *error* pada *hidden layer* (j = 1, 2, 3…. m)

$$\delta net_j = \sum_{k=1}^{m} \delta_k\, w_{jk} \quad (2.7)$$

Mengitung nilai δ_j

$$\delta_j = \delta_{\text{net}_j} . f'\left(z_{net_j}\right) = \delta_{\text{net}_j} . z_j(1 - z_j) \quad (2.8)$$

Menghitung nilai Δv_{ij} yang nantinya akan digunakan untuk melakukan *update* bobot v_{ij} yang baru

$$\Delta v_{ji} = \alpha \delta_j x_i \quad (2.9)$$

Keterangan:

δnet_j = Faktor perbaikan pada nilai bobot di z_j

δ_k = Faktor perbaikan nilai bobot pada *output layer*

w_{jk} = Nilai bobot antara *hidden layer* dengan *output layer*

δj = Faktor perbaikan nilai bobot pada *hidden layer*

z_j = Nilai sinyal pada *hidden layer*

Δv_{ji} = Perubahan bobot antara *input layer* dengan *hidden layer*

α = Nilai *learning rate*

x_i = Nilai sinyal pada *input layer*

Fase 3 : *Weight Update*

9. Langkah 8 : Menghitung semua perubahan bobot pada setiap *layer*

Persamaan untuk meng-*update* bobot antara *input layer* dengan *hidden layer*

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2.10)$$

Keterangan:

V_{ij} (baru) = bobot baru antara *input layer* dengan *hidden layer*

V_{ij} (lama) = bobot lama antara *input layer* dengan *hidden layer*

Δv_{ij} = Nilai perubahan bobot antara *input layer* dengan *hidden layer*

Persamaan untuk meng-*update* bobot antara *hidden layer* dengan *output layer*

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.11)$$

Keterangan:

W_{jk} (baru) = Bobot baru antara *hidden layer* dengan *output layer*

W_{jk} (lama) = Bobot lama antara *hidden layer* dengan *output layer*

Δw_{jk} = Nilai perubahan bobot antara *hidden layer* dengan *output layer*

10. Langkah 9 : Memeriksa kondisi berhenti sudah terpenuhi atau belum. Berikut ini beberapa kondisi berhenti yang digunakan, adalah:
    a. Menentukan kondisi berhenti berdasarkan nilai *error* yang sudah ditentukan di awal.
    b. Menentukan kondisi berhenti berdsarkan jumlah iterasi yang sudah ditentukan di awal.

BAB 3 METODOLOGI

Pada bab Metodologi dijelaskan beberapa tindakan yang dilakukan oleh peneliti untuk melakukan penelitian Implementasi Metode Jaringan Saraf Tiruan *Backpropagation* Pada Prediksi *Payload* 4G Di Telkomsel. Tindakan tersebut akan dibagi menjadi beberapa sub bab yaitu tipe penelitian, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, data penelitian, teknik analisis data dan implementasi algoritme.

3.1 Tipe Penelitian

Tipe penelitian yang dilakukan pada penelitian Implementasi Metode Jaringan Saraf Tiruan *Backpropagation* Pada Prediksi *Payload* 4G di Telkomsel adalah non–implementatif dan sifatnya analitik. Proses penelitian ini berfokus pada hasil analisis yang menghasilkan sebuah solusi dari permasalahan yang berupa prediksi *payload* 4G di Telkomsel dengan menggunakan jenis penelitian kuantitatif.

3.2 Strategi Penelitian

Strategi penelitian yang dilakukan untuk menyelesaikan masalah dalam penelitian ini adalah strategi penelitian yang bersifat experimental dimana peneliti akan melakukan percobaan antara variabel satu dengan variabel yang lain. Sehingga didapatkan suatu variabel yang paling optimal dan nantinya menjadi objek khusus yang harus diteliti.

3.3 Partisipan Penelitian

Partisipan yang terlibat langsung dalam penelitian ini adalah pegawai Telkomsel yang ada di Jember. Karakteristik dari pegawai yang terlibat langsung dalam penelitian ini adalah pegawai yang mengetahui tentang data *payload* 4G yang ada di Kabupaten Jember atau pegawai yang mengetahui tentang pentingnya *payload* terhadap suatu jaringan telepon.

3.4 Lokasi Penelitian

Lokasi penelitian yang dipakai untuk menyelesaikan masalah prediksi *payload* 4G dengan menggunakan metode jaringan saraf tiruan *backpropagation* adalah di tempat *Radio, Transport and Power Operation* (RTPO) Jember.

3.5 Teknik Pengumpulan Data

Peniliti menggunakan teknik pengumpulan data berupa observasi di *Radio, Transport and Power Operation* (RTPO) Jember. Peneliti meng-observasi masalah yang terjadi di RTPO Jember. Setelah mendapatkan suatu masalah kemudian dilakukan analisis terhadap masalah tersebut dan berdiskusi dengan pegawai Telkomsel untuk solusi yang tepat untuk masalah tersebut.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

3.6 Data Penelitian

Data yang didapatkan dari pegawai Telkomsel Jember adalah data *payload traffic* dari bulan Desember 2017 sampai bulan Januari 2018 dengan data perhari dan bertipe *time series*. Data tesebut berjumlah 62 data yang akan dibuat sebuah pola dengan 4 data masukan dan 1 data target sehingga terbentuk 58 pola data.

3.7 Teknik Analisis Data

Data yang didapatkan akan diproses oleh peneliti dan kemudian dilakukan analisis data dengan menggunakan metode *backpropagation* pada pengujian data sehingga didapatkan data yang cocok untuk digunakan sebagai data latih atau data uji. Kemudian dilakukan juga pengujian *learning rate* agar didapatkan nilai yang paling optimal.

3.8 Implementasi Algoritme

Algoritme yang digunakan adalah Jaringan Saraf Tiruan *Backpropagation*. Secara umum algoritme *backpropagation* bekerja untuk mempelajari pola data latih yang kemudian diterapkan pada data uji. Namun pada kasus prediksi, metode *backpropagation* akan memperlajari pola data latih yang berupa data hari yang sebelumnya dan kemudian melakukan proses pelatihan dan menghasilkan sebuah prediksi dengan menggunakan parameter–parameter yang sudah ditentukan.

BAB 4 PERANCANGAN

Pada Bab 4 dibahas beberapa cara untuk menyelesaikan masalah prediksi *payload* 4G di Telkomsel Jember dengan menggunakan metode jaringan saraf tiruan *backpropagation* dengan menunjukkan hasil perhitungan manual dan dilengkapi gambaran *flowchart* serta gambaran antarmuka yang diterapkan pada aplikasi yang dibuat.

4.1 Deskripsi Masalah

Peneliti mengangkat masalah yang ada di PT. Telkomsel Jember berupa permasalahan prediksi jumlah *payload* 4G yang digunakan pada tiap *site* yang ada di Jember sehingga para pegawai Telkomsel dapat mengetahui jumlah *payload* 4G yang digunakan di hari berikutnya.

Data yang digunakan adalah data per hari jumlah *payload* 4G selama dua bulan. Kemudian data tersebut dibentuk dalam sebuah pola dimana data masukan ke sistem yaitu data pada hari ke-1 hingga hari ke-4 dan data hari ke-5 merupakan data keluaran yang termasuk dari data target yang diinginkan. Selanjutnya data hari ke-2 hingga hari ke-5 dijadikan sebagai data masukan dan data hari ke-6 sebagai data keluaran, begitu seterusnya hingga terbentuk 10 pola. Data yang sudah membentuk 10 pola tersebut akan dilakukan proses pelatihan dengan menggunakan metode *Backpropagation* agar dapat melakukan prediksi pada hari selanjutnya.

Contoh dari data yang sudah terbentuk 10 pola yang digunakan untuk proses pelatihan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data Jumlah Payload 4G

Pola	*x1*	*x2*	*x3*	*x4*	Target
1	1628137	1901409	1625850	1387557	1543720
2	1901409	1625850	1387557	1543720	1422201
3	1625850	1387557	1543720	1422201	1198254
4	1387557	1543720	1422201	1198254	1591217
5	1543720	1422201	1198254	1591217	1647456
6	1422201	1198254	1591217	1647456	1741562
7	1198254	1591217	1647456	1741562	1506706
8	1591217	1647456	1741562	1506706	1685361
9	1647456	1741562	1506706	1685361	1612741

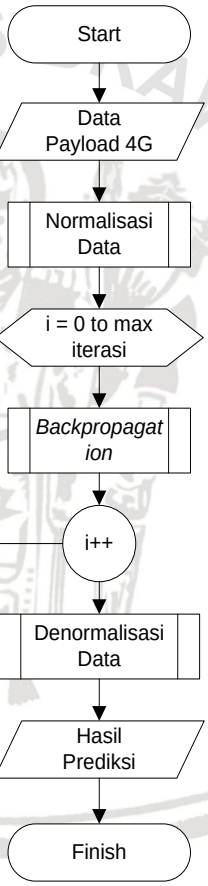
repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

10	1741562	1506706	1685361	1612741	1508271

Data *x1,x2,x3* dan *x4* adalah data masukan yang akan diproses sebagai data latih dan target adalah data yang diinginkan kemudia dibandingkan dengan hasil yang didapatkan pada proses pelatihan untuk menentukan bobot baru yang akan menghasilkan nilai target yang baru agar mendapatkan *error* yang lebih kecil.

4.2 Desain Algoritme

Desain algoritme yang akan dilakukan pada proses di dalam aplikasi dapat digambarkan dalam bentuk *flowchart*. Desain *flowchart* dapat dilihat pada Gambar 4.1.



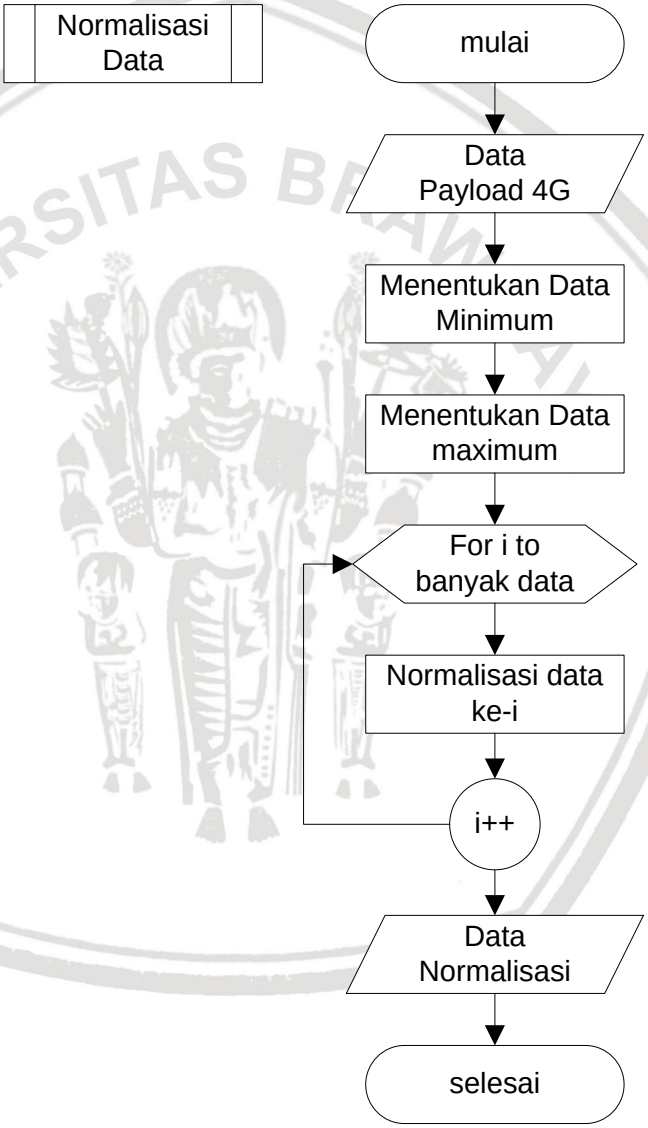
Gambar 4.1. *Flowchart* Desain Algoritme

Gambar 4.1 adalah desain *flowchart* program yang dibuat. Pada proses pertama yang akan dilakukan oleh aplikasi adalah memasukan data *Payload* 4G yang sudah didapatkan dari PT. Telkomsel. Data jumlah *Payload* 4G ini yang nantinya akan diprediksi berdasarkan hari. Sebelum melakukan proses pelatihan *backpropagation* data harus melalui sebuah proses normalisasi data terlebih dahulu agar rentang data tidak terlalu jauh. Setelah data dinormalisasi maka data

akan melalui proses pelatihan *backpropagation* agar mendapatkan nilai bobot yang optimal sehingga dapat melakukan proses prediksi dengan baik dan mendapatkan nilai *error* yang kecil. Setelah proses *backpropagation* selesai maka data harus melalui proses denormalisasi data terlebih dahulu agar mendapatkan data yang sesungguhnya. Karena pada proses normalisasi data, data akan bernilai antar 0 sampai 1. Dengan melakukan proses denormalisasi data, maka didapatkan sebuah data hasil prediksi.

4.2.1 Normalisasi Data

Proses normalisasi data dapat dilihat pada Gambar 4.2.



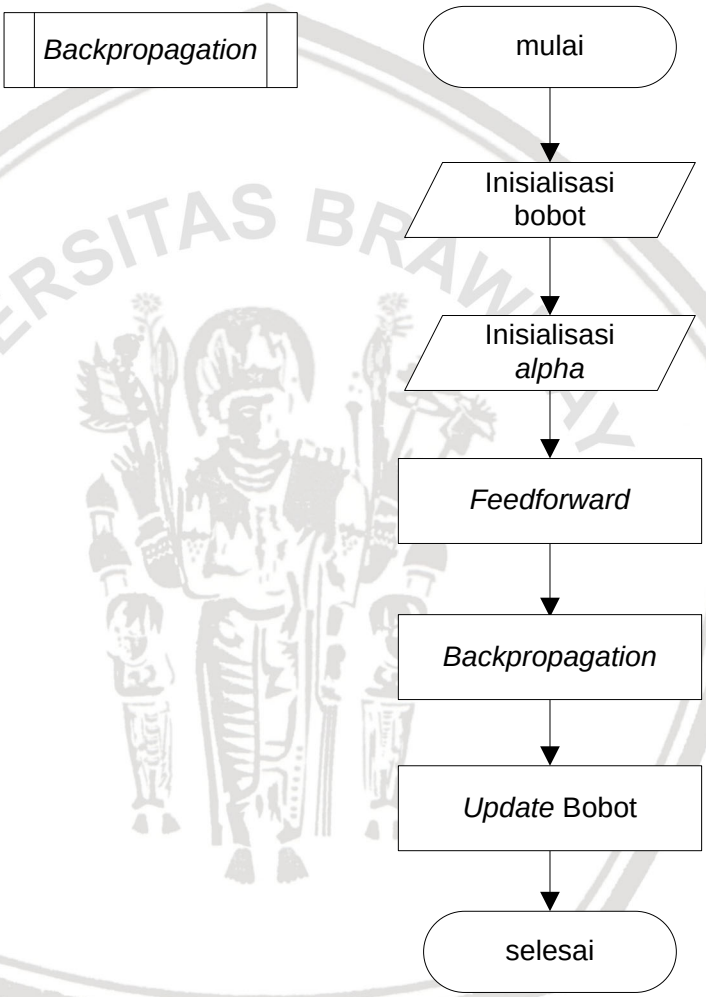
Gambar 4.2 *Flowchart* Normalisasi Data

Proses normalisasi data digunakan untuk merubah data menjadi data yang memiliki rentang nilai antara 0 sampai 1, agar data yang akan dilatih dengan algortime *backpropagation* dapat menghasilkan data yang optimal. Proses

normalisasi data, diawali dengan memasukan data jumlah *Payload* 4G yang kemudian dicari data yang paling kecil dan paling besar. Setelah menemukan data yang paling kecil dan paling besar, baru data jumlah *Payload* 4G dapat dilakukan normalisasi sebanyak data yang ada. Hasil yang didapatkan akan berupa data yang memiliki rentang nilai antara 0 sampai dengan 1.

4.2.2 *Backpropagation*

Algoritme *Backpropagation* dapat digambarkan dalam sebuah gambar *flowchart* yang dapat dilihat pada Gambar 4.3.



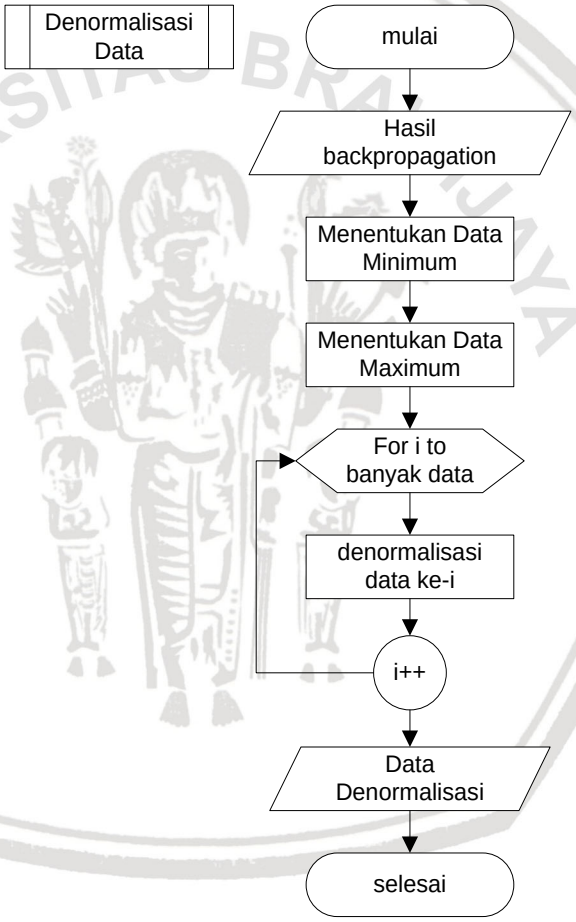
Gambar 4.3 *Flowchart* Algoritme Backpropagation

Algoritme *backpropagation* memiliki tiga proses, yaitu proses *feedforward, backpropagation* dan *update* bobot. Sebelum melalui tiga proses tersebut, harus menginisialisasikan bobot secara random yang memiliki nilai dengan rentang 0 sampai 1 dan juga menentukan nilai *learning rate* yang juga memiliki nilai rentang antara 0 sampai dengan 1. Setelah inisialisasi bobot dan *learning rate* juga sudah ditentukan maka proses pertama yang dilakukan adalah *feedforward*, pada proses *feedforward* akan menghasilkan sebuah nilai *output* yang memiliki simbol *y*. Nilai

y tersebut merupakan data prediksi, namun nilai *y* masih memiliki rentang nilai antara 0 sampai dengan 1, untuk menghasilkan nilai prediksi yang sesungguhnya harus melakukan proses denormalisasi data yang akan dilakukan nanti setelah algortime *Backpropagation* sudah selesai pada iterasi tertentu. Proses selanjutnya yakni proses *backpropagation* yang berisi perhitungan *error* yang dihasilkan dari proses *feedforward*, setelah perhitungan *error* didapatkan baru proses *update* bobot data dilakukan agar dapat memperbarui nilai bobot yang sudah ditentukan secara random di awal. Setelah algoritme *bakcpropagation* sudah selesai maka untuk menghasilkan nilai prediksi yang sesungguhnya harus melakukan proses denormalisasi data.

4.2.3 Denormalisasi Data

Proses denormalisasi data dapat dilihat pada Gambar 4.4.



Gambar 4.4. Proses Denormalisasi Data

Proses denormalisasi data digunakan untuk mendapatkan nilai yang sesungguhnya. Proses denormalisasi data diawali dengan memasukan nilai *y* yang dihasilkan dari algoritme *backpropagation.* Kemudian dicari data yang paling kecil dan paling besar baru dapat dilakukan proses denormalisasi data sebanyak data yang sudah ditentukan sebelumnya.

4.3 Perhitungan Manual Algoritme Backpropagation

Langkah pertama yang dilakukan adalah menginisialisasikan bobot yang terdiri dari bobot *v*, bobot *w* dan bobot bias. Bobot *v* adalah bobot antara *node input* dengan *node hidden layer*, sedangkan bobot *w* adalah bobot antara *node hidden layer* dengan *node output layer*. Penginisialisasian bobot akan ditentukan antara 0 sampai 1.

Tabel 4.2 Inisialisasi Bobot v

v	*z1*	*z2*	*z3*	*z4*	*z5*	*z6*
x1	0,888437	0,878039	0,731519	0,251768	0,287093	0,964413
x2	0,272458	0,765133	0,320558	0,790984	0,980435	0,555533
x3	0,581064	0,457123	0,844291	0,248526	0,318666	0,626793
x4	0,598497	0,829659	0,743909	0,916284	0,979203	0,445629
bias	0,506947	0,968992	0,319966	0,555513	0,930191	0,032458

Tabel 4.3 Inisialisasi Bobot w

w	*z1*	*z2*	*z3*	*z4*	*z5*	*z6*	bias
y	0,204705	0,491108	0,639891	0,19927	0,682498	0,206802	0,717883

Setelah menginisialisasikan bobot v dan bobot w maka proses selanjutnya adalah proses *feedforward,* proses ini di mulai dari *node input* menuju *node output.* Sebelum melakukan proses *feedforward,* data yang sudah terbentuk 10 pola tersebut harus dinormalisasi terlebih dahulu agar jarak antar data tidak terlalu jauh.

$$x' = \left(0{,}8 * \frac{x - \min value}{\max value - \min value}\right) + 0{,}1 \quad (4.1)$$

$$x' = \left(0{,}8 * \frac{1628136{,}56 - 1198254{,}112}{2274636{,}488 - 1198254{,}112}\right) + 0{,}1$$

Tabel 4.4 Data Normalisasi

Pola	*x1*	*x2*	*x3*	*x4*	Target
1	0,419501662	0,622606021	0,417802311	0,240695833	0,356760752
2	0,622606021	0,417802311	0,240695833	0,356760752	0,266444441
3	0,417802311	0,240695833	0,356760752	0,266444441	0,1
4	0,240695833	0,356760752	0,266444441	0,1	0,392061599
5	0,356760752	0,266444441	0,1	0,392061599	0,433860206
6	0,266444441	0,1	0,392061599	0,433860206	0,503803249
7	0,1	0,392061599	0,433860206	0,503803249	0,329250838
8	0,392061599	0,433860206	0,503803249	0,329250838	0,462032362
9	0,433860206	0,503803249	0,329250838	0,462032362	0,408059109

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

10	0,503803249	0,329250838	0,462032362	0,408059109	0,330414136

Tabel 4.4 adalah data hasil normalisasi dari data jumlah *payload* 4G yang nantinya akan dijadikan sebagai data latih pada proses *feedforward*. Setelah data sudah dinormalisasi maka proses *feedforward* dapat dilakukan, langkah pertama untuk melakukan proses *feedforward* adalah menghitung nilai *z_net*. Nilai *z_net* merupakan nilai dari *hidden node* yang berada di antara *input node* dengan *output node*. Perhitungan nilai *z_net* dapat dilakukan dengan cara menjumlahkan semua perkalian antara *input node* dengan bobot yang sudah diinisialisasikan pada proses sebelumnya.

$$z_net_j = v_{j0} + \sum_{i=1}^{n} x_i . v_{ji} \quad (4.2)$$

$$z_net_1 = 0.506947118\ (0{,}419501662 * 0{,}88843717) + (0{,}622606021 * 0{,}272457949) + (0{,}417802311 * 0{,}581063503) + (0{,}240695833 * 0{,}598496761) = 1{,}436107$$

Untuk hasil yang lebih lengkap dari perhitung *z_net* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Nilai Z_netj

z_net1	*z_net2*	*z_net3*	*z_net4*	*z_net5*	*z_net6*
1,436107	2,20439	1,358223	1,477981	2,029882	1,152046

Setelah semua nilai *z_net* diketahui, maka nilai *z_net* tersebut harus dihitung kembali dengan menggunakan fungsi aktifasi yang nantinya akan menjadi nilai *z_j*.

$$z_j = f(z_net_j) = \frac{1}{1+exp^{(-z_net_j)}} \quad (4.3)$$

$$z_1 = \frac{1}{1 + exp^{(-1{,}436107298)}} = 0{,}807851$$

Seluruh nilai dari *Zj* dapat dilihat pada Tabel 4.6

Tabel 4.6 Nilai Zj

z1	*z2*	*z3*	*z4*	*z5*	*z6*
0,807851	0,900643	0,795471	0,814267	0,883899	0,759884

Setelah memperoleh nilai *zj*, maka langkah selnjutnya yaitu mencari nilai *output* yaitu nilai *y_net*. *Y_net* adalah nilai *output* yang masih belum dihitung dengan menggunakan fungsi aktifasi. Nilai *y_net* dapat didapatkan dengan cara menjumlahkan semua perkalian antara nilai *zj* yang termasuk nilai *node hidden layer* dengan bobot *zj* yang sudah diinisialisasikan pada perhitungan sebelumnya.

$$y_net_k = w_{k0} + \sum_{j=1}^{p} z_j . w_{kj} \quad (4.4)$$

$$\begin{aligned} y_net_1 = {} & 0{,}717883491 + (0{,}204704501 * 0{,}80785112) \\ & + (0{,}491107925 * 0{,}900643039) \\ & + (0{,}639890894 * 0{,}795470817) \\ & + (0{,}199270334 * 0{,}814267497) \\ & + (0{,}682497741 * 0{,}883898927) \\ & + (0{,}206802392 * 0{,}759884399) = 2{,}757246005 \end{aligned}$$

Setelah nilai *y_net* diketahui, kemudian menghitung nilai *y_net* tersebut dengan menggunakan fungsi aktifasi agar mendapatkan nilai *y*.

$$y_k = f(y_net_k) = \frac{1}{1+exp^{(-y_net_k)}} \quad (4.5)$$

$$y_k = \frac{1}{1 + exp^{(-2{,}757246005)}} = 0{,}940321274$$

Dari proses *feedforward* dapat diketahui bahwa nilai *output* yang dihasilkan adalah sebesar 0,940321274. Dengan diketahuinya nilai *y* sebagai nilai *output*, maka proses *feedforward* sudah selesai. Proses selanjutnya yaitu proses *backpropagation*, dimana proses ini akan dimulai dari *output* menuju ke *input*. Untuk memulai proses *backpropagation* hal yang harus dilakukan adalah menghitung nilai faktor *error* atau bisa ditulis dengan lambang *δk*.

$$\delta_k = (t_k - y_k) f'(y_net_k) = (t_k - y_k) y_k (1 - y_k) \quad (4.6)$$

$$\begin{aligned} \delta = {} & (0{,}356760752 - 0{,}940321274) * 0{,}940321274 \\ & * (1 - 0{,}940321274) = -0{,}032747767 \end{aligned}$$

Langkah kedua pada proses *backpropagation* yaitu menghitung perubahan bobot w_{kj} atau dapat ditulis dengan lambang Δw_{kj}. Perhitungan perubahan bobot ini digunakan untuk meng-*update* bobot, agar mendapatkan bobot yang lebih optimal. Pada perhitungan Δw_{kj} peneliti menggunakan nilai *learning rate* sebesar 0,1 . nilai Δw_{kj} dapat diketahui dengan cara mengalikan *learning rate*, δ_k dan z_j.

$$\Delta w_{kj} = \propto . \delta_k . z_j \quad (4.7)$$

$$\Delta w_{11} = 0{,}1 * -0{,}032747767 * 0{,}807851120 = -0{,}000529106$$

Untuk melihat hasil keseluruhan dari nilai Δw_{kj} dapat dilihat pada Tabel 4.7.

Tabel 4.7 Nilai Δw_{kj}

Δw10	*Δw11*	*Δw12*	*Δw13*	*Δw14*	*Δw15*	*Δw16*
-0,00065	-0,00053	-0,00059	-0,00052	-0,00053	-0,00058	-0,000498

Setelah nilai Δw_{kj} diketahui secara keseluruhan, kemudian menghitung nilai δ pada *hidden layer* atau dapat ditulis dengan simbol δ_net_j. Untuk menghitung δ_net_j dapat dilakukan dengan cara mengalikan δ_k dengan Δw_{kj}.

$$\delta_net_j = \sum_{k=1}^{m} \delta_k . \Delta w_{kj} \quad (4.8)$$

$$\delta_net_1 = -0{,}032747767 * -0{,}000529106 = 0{,}000017327054197621 4$$

Nilai δ_net_j secara keseluruhan dapat dilihat pada Tabel 4.8.

Tabel 4.8 Nilai δ_net_j

δ_netj1	δ_netj2	δ_netj3	δ_netj4	δ_netj5	δ_netj6
1,73271E-05	1,93173E-05	1,70615E-05	1,74647E-05	1,89582E-05	1,62982E-05

Dengan mengetahui nilai δ_net_j secara keseluruhan, maka dapat diketahui pula nilai δ_j dengan cara menghitung nilai δ_net_j dengan fungsi sigmoid.

$$\delta_j = \delta_net_j . f'(z_net_j) = \delta_net_j . z_j . (1 - z_j) \quad (4.9)$$

$$\delta_1 = 0{,}000017327054197621 4 * 0{,}807851 * (1 - 0{,}807851) = 2{,}68963855743957E-06$$

Nilai δ_j secara keseluruhan dapat dilihat pada Tabel 4.9

Tabel 4.9 Nilai δ_j

δ_j1	δ_j2	δ_j3	δ_j4	δ_j5	δ_j6
2,68964E-06	1,72861E-06	2,77586E-06	2,64129E-06	1,94552E-06	2,97378E-06

Langkah selanjutnya adalah menghitung nilai perubahan bobot *delta v* yang dapat ditulis dengan simbol Δv_{ji}. Δv_{ji} digunakan untuk meng-*update* bobot *v* yang baru. Nilai dari Δv_{ji} dapat diketahui dengan cara mengalikan α, δ_j serta x_i.

$$\Delta v_{ji} = \alpha . \delta_j . x_i \quad (4.10)$$

$$\Delta v_{11} = 0.1 * 2{,}68963855743957 * 0{,}419501662297748 = 2{,}25661569165204$$

Nilai Δv_{ji} secara keseluruhan dapat dilihat pada Tabel 4.10

Tabel 4.10 Nilai Δv_{ji}

Δv	i1	i2	i3	i4	bias
j1	2,26E-08	3,35E-08	2,25E-08	1,29E-08	5,38E-08
j2	1,45E-08	2,15E-08	1,44E-08	8,32E-09	3,46E-08
j3	2,33E-08	3,46E-08	2,32E-08	1,34E-08	5,55E-08
j4	2,22E-08	3,29E-08	2,21E-08	1,27E-08	5,28E-08
j5	1,63E-08	2,42E-08	1,63E-08	9,37E-09	3,89E-08
j6	2,5E-08	3,7E-08	2,48E-08	1,43E-08	5,95E-08

Perhitungan nilai Δv_{ji} merupakan perhitungan terakhir dari proses *backpropagation.* Proses selanjutnya setelah proses *backpropagation* adalah proses *update* bobot. Bobot pertama yang di*update* adalah bobot *w*. Untuk meng*update* bobot *w* dapat dilakukan dengan cara menambahkan bobot awal dengan nilai perbuahan bobot yang sudah diketahui pada perhitungan Δw_{kj}.

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (4.11)$$

$$w_{11}(baru) = 0{,}204704500875757 + (-0{,}000529106) = 0{,}204175394456473$$

Untuk melihat nilai $w_{kj}(baru)$ secara keseluruhan dapat dilihat pada Tabel 4.11.

Tabel 4.11 Nilai $w_{kj}(baru)$

w	*z1*	*z2*	*z3*	*z4*	*z5*	*z6*	bias
y	0,717229	0,204175	0,490518	0,63937	0,198737	0,681919	0,206305

Setelah melakukan *update* bobot $w_{kj}(\text{baru})$, baru kemudian meng-*update* bobot $v_{ji}(\text{baru})$. Untuk melakukan *update* bobot $v_{ji}(\text{baru})$ dapat dilakukan dengan cara menambahkan bobot $v_{ji}(\text{lama})$ dengan nilai perubahan bobot yang sudah diketahui pada perhitungan sebelumnya.

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (4.12)$$

$$v_{11}(baru) = 0{,}888437170302854 + 2{,}25661569165204 = 0{,}888437192869011$$

Untuk melihat nilai $v_{ji}(\text{baru})$ secara keseluruhan dapat dilihat pada Tabel 4.12.

Tabel 4.12 Nilai $v_{ji}(baru)$

v	x1	x2	x3	x4	bias
z1	0,888437193	0,272457982	0,581063525	0598496774	0,506947172
z2	0,878038572	0,765133235	0,457122937	0,829659188	0,968992292
z3	0,731518777	0,320558218	0,844290557	0,743908883	0,31996646
z4	0,251768358	0,790983775	0,248525912	0,916283551	0,555512704
z5	0,287093142	0,980435445	0,318666496	0,979202538	0,930191027
z6	0,964413106	0,555532856	0,626793131	0,445629442	0,032458173

Proses *feedforward, backpropagation* dan *update* bobot dilakukan berulang–ulang sampai pada iterasi tertentu. Namun disini peneliti melakukan iterasi pada perhitungan manual sebanyak tiga kali. Pada iterasi ketiga didapatkan bobot $w_{kj}(baru)$ dan bobot $v_{ji}(baru)$ yang dapat dilihat secara berturut–turut pada Tabel 4.13 dan Tabel 4.14.

Tabel 4.13 Bobot Akhir w_{kj}

w	z1	z2	z3	z4	z5	z6	bias
y	0,189751	0,474446	0,625229	0,184315	0,666199	0,193172	0,698938

Tabel 4.14 Bobot Akhir v_{ji}

v	x1	x2	x3	x4	bias
z1	0,888437831	0,272458593	0,581064103	0,598497348	0,506948924
z2	0,878039021	0,765133663	0,457123348	0,829659592	0,968993543
z3	0,731519439	0,320558848	0,844291151	0,743909475	0,319968266
z4	0,251769006	0,790984384	0,24852649	0,916284123	0,555514461
z5	0,287093652	0,980435921	0,318666951	0,979202986	0,930192418
z6	0,964413814	055553353	0,626793769	0,445630078	0,032460103

Setelah bobot w_{kj} dan bobot v_{ji} diketahui maka nilai y yang sebagai nilai *output* dapat diketahui dengan menggunakan proes *feedforward*. Nilai y adalah hasil prediksi dari proses algoritme *backpropagation*. Hasil dari prediksi dapat dilihat pada Tabel 4.15.

Tabel 4.15 Hasil Nilai *y*

Nilai *y*
0,936724945
0,936491536
0,931159881
0,925098858
0,928971522
0,929637371
0,932745721
0,934900783
0,936061385
0,935380918

Nilai *y* adalah nilai yang masih belum dilakukan proses perhitungan denormalisasi. Untuk dapat mengetahui hasil prediksi, harus dilakukan proses denormalisasi data terlebih dahulu karena sebelum dilakukan denormalisasi data, data masih berupa data normalisasi dimana data memiliki rentang 0 sampai 1.

$$x'' = \frac{(\max value - \min value) * (x' - 0{,}1)}{0{,}8} + \min\ value \quad (4.13)$$

$$x'' = \frac{(2274636{,}488 - 1198254{,}112) * (0{,}936724945200158 - 0{,}1)}{0{,}8} + 1198254{,}112 = 2324049{,}09271627$$

Hasil denormalisasi secara keseluruhan dapat dilihat pada Tabel 4.16

Tabel 4.16 Hasil Denormalisasi Nilai *Y*

Nilai *y*
2324049,093
2323735,046
2316561,421
2308406,449
2313617,032
2314512,918
2318695,134
2321594,722
2323156,287
2322240,734

Setelah mendapatkan nilai *y* dengan cara denormalisasi data, maka hasil prediksi dengan menggunakan algoritme *backpropagation* dapat diketahui.

Berikut ini Tabel 4.17 menunjukkan hasil aktual dan hasil prediksi dari algoritme *backpropagation*, serta Tabel 4.18 menunjukkan hasil *error* dari algoritme *backpropagation* dengan cara menghitung nilai MSE.

Tabel 4.18 Hasil Aktual dengan Hasil Prediksi

Tanggal	Aktual	Prediksi
05/12/2017	1543720,048	2324049,093
06/12/2017	1422201,44	2323735,046
07/12/2017	1198254,112	2316561,421
08/12/2017	1591216,56	2308406,449
09/12/2017	1647455,664	2313617,032
10/12/2017	1741562,488	2314512,918
11/12/2017	1506706,064	2318695,134
12/12/2017	1685360,68	2321594,722
13/12/2017	1612740,856	2323156,287
14/12/2017	1508271,256	2322240,734

Tabel 4.19 Hasil *Error* Dengan MSE

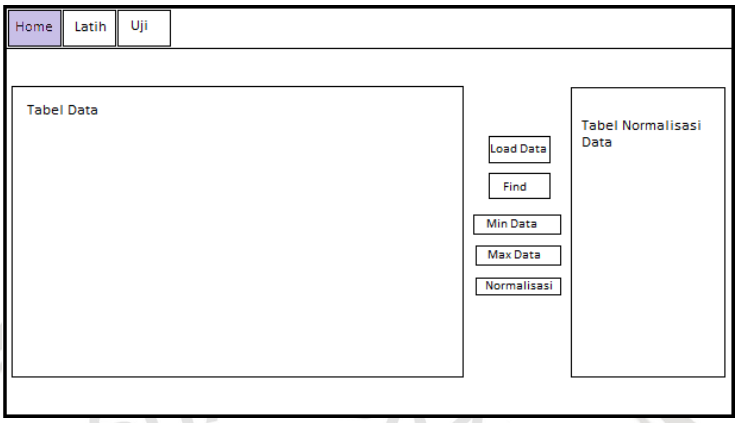
Tanggal	aktual	Nilai *y*	Selisih	Selisih Kuadrat
05/12/2017	0,356760752	0,936724945	0,579964193	0,336358465
06/12/2017	0,266444441	0,936491536	0,670047096	0,448963111
07/12/2017	0,1	0,931159881	0,831159881	0,690826747
08/12/2017	0,392061599	0,925098858	0,533037259	0,284128719
09/12/2017	0,433860206	0,928971522	0,495111316	0,245135215
10/12/2017	0,503803249	0,929637371	0,425834122	0,181334699
11/12/2017	0,329250838	0,932745721	0,603494883	0,364206073
12/12/2017	0,462032362	0,934900783	0,47286842	0,223604543
13/12/2017	0,408059109	0,936061385	0,528002277	0,278786404
14/12/2017	0,330414136	0,935380918	0,604966782	0,365984807
MSE =				0,341932878

4.4 Perancangan Antarmuka

Pada subbab perancangan antarmuka dijelaskan bagaimana bentuk dari aplikasi prediksi jumlah *payload* 4G di Telkomsel Jember yang akan dibuat. Pembuatan aplikasi prediksi jumlah *payload* 4G di Telkomsel Jember dibuat dengan menggunakan empat halaman yang terdiri dari halaman *home,* latih, uji dan halaman ramal.

4.4.1 Perancangan Halaman *Home*

Pada Halaman *home* berisi tabel data yang akan diprediksi menggunakan algoritme *backpropagation* dan juga berisi tabel data yang menunjukkan normalisasi data yang telah dipilih. Gambar perancangan halaman *home* dapat dilihat pada Gambar 4.1.



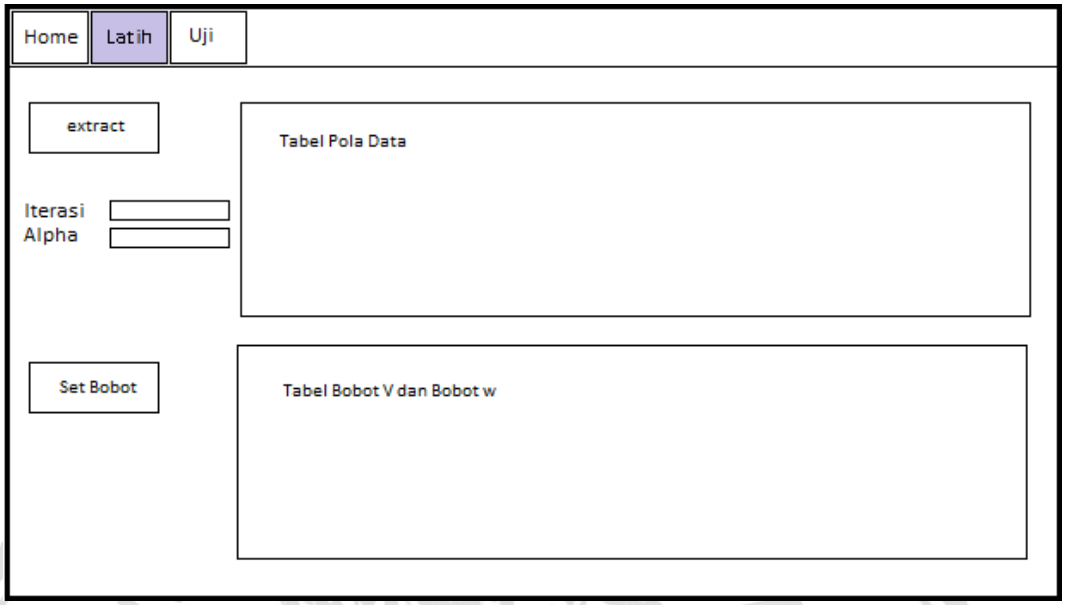
Gambar 4.5 Perancangan Halaman *Home*

Keterangan:

1. Tabel Data digunakan untuk menunjukkan semua data yang akan diprediksi,
2. *Button load data* digunakan untuk memasukan data,
3. Button *find* digunakan untuk mencari data berdasarkan *site* yang nantinya akan digunakan untuk proses pelatihan dan pengujian,
4. *Button* min data digunakan untuk mencari data minimum yang nantinya akan digunakan pada proses normalisasi data,
5. *Button* max data digunakan untuk mencari data maksimum yang nantinya akan digunakan pada proses normalisasi data,
6. *Button* normalisasi digunakan untuk menormalisasi data yang sudah dipilih, serta
7. Tabel normalisasi digunakan untuk menampilkan hasil normalisasi data.

4.4.2 Perancangan Halaman Latih

Pada halaman latih terdapat proses pelatihan data yang menampilkan hasil inisialisasi bobot *v* dan *w* yang dapat dilihat pada Gambar 4.2.



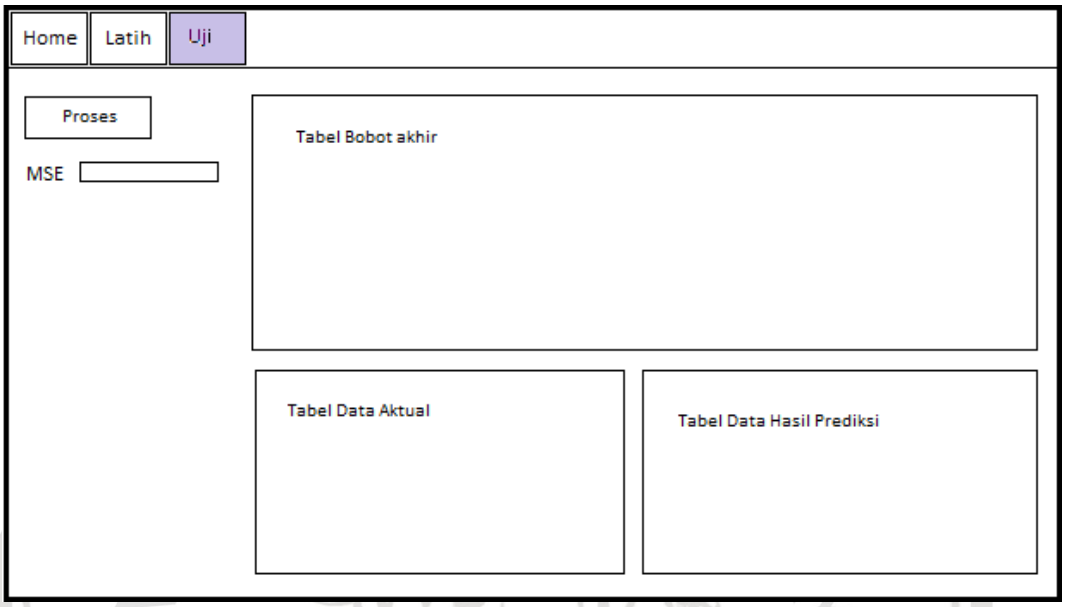
Gambar 4.6 Perancangan Halaman Latih

Keterangan:

1. *Button Extract* digunakan untuk memasukan data normalisasi ke dalam Tabel Pola Data dengan menyesuaikan dengan pola yang sudah ditentukan,
2. *Button* set bobot digunakan untuk menganasialisasikan bobot awal yang nantinya akan digunakan untuk proses pelatihan,
3. *TextBox* iterasi dan alpha digunakan untuk memasukan jumlah iterasi dan *alpha* yang akan digunakan pada proses pelatihan.

4.4.3 Perancangan Halaman Uji

Pada perancangan halaman uji terdapat proses pengujian data menghasilkan nilai bobot akhir dari bobot v dan bobot w, serta menampilkan hasil prediksi dan hasil *error* dengan menggunakan MSE. Gambar perancangan halaman uji dapat dilihat pada Gambar 4.3.



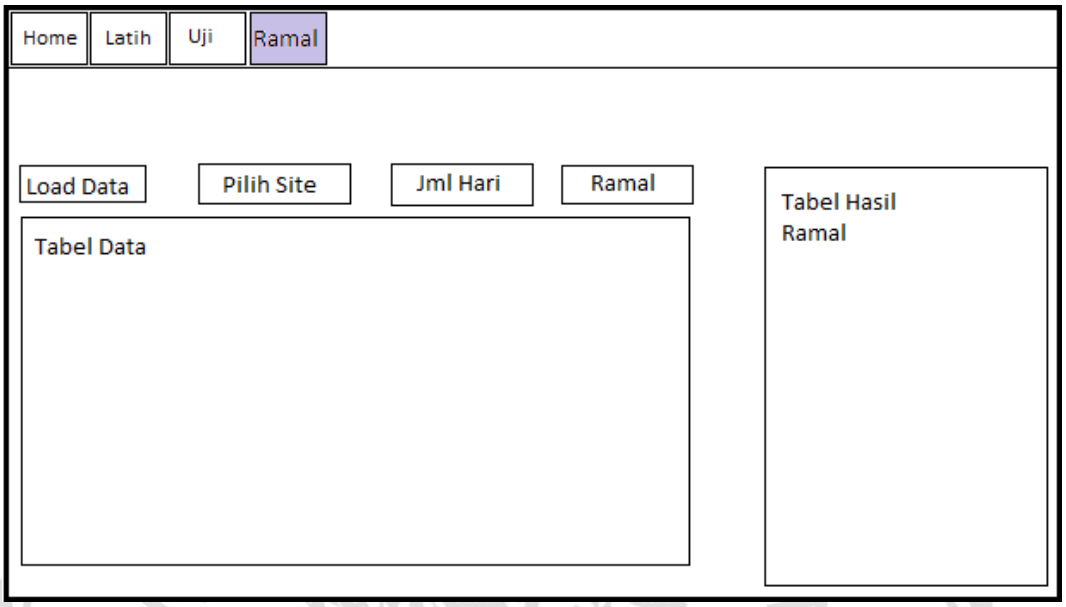
Gambar 4.7 Perancangan Halaman Uji

Keterangan:

1. *Button* proses digunakan untuk memproses data yang nantinya akan menampilkan hasil *error*, bobot akhir dan hasil prediksi,
2. MSE akan menampilkan hasil *error* yang terjadi dan ditampilkan pada *textbox* MSE,
3. Tabel bobot akhir berguna untuk menampilkan bobot akhir yang digunakan untuk melakukan prediksi
4. Tabel data aktual dan tabel data hasil prediksi untuk menampilkan nilai target yang ingin diperoleh dan hasil prediksi yang sudah diproses oleh algoritme *backpropagation*.

4.4.4 Perancangan Halaman Ramal

Pada halaman ramal terdapat sebuah proses yang akan digunakan oleh *user* untuk melakukan prediksi pada hari yang diinginkan oleh *user*. Gambar perancangan halaman ramal dapat dilihat pada Gambar 4.4.



Gambar 4.8 Perancangan Halaman Ramal

Keterangan:

1. *Button Load Data* digunakan untuk memasukan data yang akan diprediksi,
2. *Button* pilih *site* digunakan untuk memilih *site* mana yang akan diprediksi,
3. Jml hari adalah sebuah *text box* yang akan diisi oleh *user* sesuai jumlah hari yang ingin diprediksi oleh *user*,
4. Button ramal digunakan untuk memproses dan menghasilkan data prediksi.

BAB 5 IMPLEMENTASI

Pada Bab Implementasi dijelaskan tentang penerapan metode *Backpropagation* pada masalah prediksi *payload* 4G yang ada di PT. Telkomsel dengan menggunakan tampilan yang sudah digambarkan pada Bab Perancangan. Bab Implementasi juga berisi penjelasan tentang *source code* yang sudah dibuat.

5.1 Implementasi Program Metode *Backpropagation*

Subbab ini menjelaskan tentang penerapan metode *Backpropagation* pada sebuah program java yang dibuat pada sebuah aplikasi Netbeans. Pada pembuatan aplikasi prediksi *payload* 4G pada PT. Telkomsel dengan menggunkaan metode *backpropagation* dibuat sebuah fungsi utama yang terdiri dari *load* data, normalisasi data, *extract* data, set bobot, proses pelatihan dan proses pengujian.

5.1.1 Proses *Load* Data

Load data digunakan untuk memasukan data latih yang akan digunakan untuk proses pelatihan agar mendapatkan nilai bobot yang optimal untuk proses pengujian. *Source code Load* data dapat dilihat pada Kode Program 5.1.

```
1  private void Load_Data() {
2      dm.setRowCount(0);
3      JFileChooser fileChooser = new JFileChooser();
4      fileChooser.setCurrentDirectory(new File(System.
5      getProperty("user.home")));
6      int result = fileChooser.showOpenDialog(this);
7      if (result == JFileChooser.APPROVE_OPTION) {
8          File file = fileChooser.getSelectedFile();
9          try {
10             BufferedReader br = new BufferedReader(new
11             FileReader(file));
12             String firstLine = br.readLine().trim();
13             String[] columnsName = firstLine.split(";");
14             DefaultTableModel model = (DefaultTableModel)
15             Tabel_Data.getModel();
16             model.addColumn("Nomer");
17             model.setColumnIdentifiers(columnsName);
18
19             // get lines from txt file
20             Object[] tableLines = br.lines().toArray();
21
```

```
22              // extratct data from lines
23              // set data to jtable model
24              for(int i = 0; i < tableLines.length; i++)
25              {
26                  String line = tableLines[i].toString()
27                  .trim();
28                  String[] dataRow = line.split(";");
29                  model.addRow(dataRow);
30              }
31
32          } catch (Exception ex) {
33               Logger.getLogger(Gui_Ramal.class.getName())
34               .log(Level.SEVERE, null, ex);
35          }
36      }
37  }
```

Kode Program 5.1 *Load* Data

Penjelasan:

1 - 6	Menginisialisasikan *file* yang dipilih dengan nama variable "file" dan *file* yang dipilih harus berekstensi .csv
8 – 16	Menyimpan data dari *file* csv ke dalam sebuah *array*, karena data bertipe csv dan akan ditampilkan dalam sebuah tabel maka antar kolom dipisahkan dengan menggunakan tanda ";"
20 - 25	Menggunkan perulangan "For" untuk membaca *array* yang sudah dibuat dan menampilkannya dalam sebuah tabel

5.1.2 Proses Normalisasi Data

Sebelum data digunakan untuk pelatihan pada metode *Backpropagation* data perlu dilakukan proses normalisasi terlebih dahulu. Proses normalisasi data digunakan agar data yang dilatih memiliki rentang nilai antara 0 sampai dengan 1, sehingga jarak antara satu data dengan data yang lainnya tidak terlalu jauh.

```
 1  private void Normalisasi_Data() {
 2      ArrayList<Double> list = new ArrayList<Double>();
 3      int x = Tabel_Data.getRowCount();
 4      for (int count = 0 ; count < x ; count++)
 5      {
 6         list.add(Double.parseDouble(Tabel_Data
 7         .getValueAt(count,4).toString()));
 8      }
 9      double max = Collections.max(list);
10      double min = Collections.min(list);
```

```
11      dm2.setRowCount(0);
12      double hasil ;
13
14      for (int count = 0 ; count < x ; count++)
15      {
16         double z = Double.parseDouble(Tabel_Data
17         .getValueAt(count, 4).toString());
18         hasil = ((0.8*(z - min))/(max - min)) + 0.1;
19         //Double[] rowData = {hasil};
20         dm2.addRow(new Object[]{Tabel_Data
21         .getValueAt(count, 0),hasil});
22      }
23
24  }
```

Kode Program 5.2 Normalisasi Data

Penjelasan:

1 - 5	Membaca tabel data dengan memasukan data ke dalam sebuah *array list* dengan menggunakan perulangan "For"
6 - 7	Mencari nilai min dan max
11 - 17	Dengan menggunakan perulangan "For" untuk menghitung normalisasi data dengan menggunakan data asli, nilai min data dan max data sebanyak data yang ditampilkan di tabel

5.1.3 Proses Extract Data

Proses *extract* data adalah proses dimana semua normalisasi data akan disusun menjadi suatu pola yang nantinya akan dijadikan sebagai data latih. Jumlah dari pola yang akan ditentukan pada proses *extract* data akan sangat mempengaruhi proses pelatihan.

```
 1  private void Extract_Data() {
 2      double table2[] = new double[Tabel_Normalisasi
 3      .getRowCount()] ;
 4      int tot_n = Tabel_Normalisasi.getRowCount();
 5      int pola_tf = Integer.parseInt(jml_pola.getText());
 6      int x;
 7      int p = Tabel_Normalisasi.getRowCount();
 8      double tes;
 9      dm3.setRowCount(0);
10      String combo = combo_data.getSelectedItem().toString();
11
12      if ("Awal".equals(combo))
13      {
```

```
14          x = 0;
15      }
16      else if ("Tengah".equals(combo))
17      {
18          x = 0;
19          pola_tf = 20;
20      }
21      else
22      {
23          x = tot_n-4-pola_tf;
24      }
25
26      for (int a = 0 ; a < p ; a++)
27      {
28          table2[a] = Double.parseDouble(Tabel_Normalisasi
29          .getValueAt(a, 2).toString());
30
31      }
32      int count1 = 0;
33      int count2 = 0;
34
35      for (int b = 0 ; b < p-4 ; b++)
36      {
37          for (int c = 1 ; c < 6 ; c++)
38          {
39              trainInputs[b][c] = table2[count2];
40              count2++;// = 1
41          }
42          trainInputs[b][0] = 1;
43          count1++;
44          count2 = count1;
45      }
46      for (int l = 0 ; l < pola_tf ; l++)
47      {
48          dm3.addRow(new Object[]{1, trainInputs[x][1]
49          ,trainInputs[x][2],trainInputs[x][3]
50          ,trainInputs[x][4],trainInputs[x][5]});x++;
51      }
52      if ("Tengah".equals(combo))
53      {
54          int m = tot_n - 4 - pola_tf;
55          for (int l = 0 ; l < pola_tf ; l++)
56          {
57              dm3.addRow(new Object[]{1, trainInputs[m][1]
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
58                ,trainInputs[m][2],trainInputs[m][3]
59                ,trainInputs[m][4],trainInputs[m][5]});m++;
60            }
61        }
62    }
```

Kode Program 5.3 *Extract* Data

Penjelasan:

1 - 8	Inisialisasi variabel
9 - 21	Penentuan kondisi untuk pengujian *k-fold validation* untuk data latih yang akan digunakan yaitu data awal, tengah dan akhir.
23 - 28	Membaca tabel normalisasi data kemudian menyimpan data tersebut ke dalam sebuah *array table2*[]
32 - 42	Menyusun *array tabel2*[] sebanyak jumlah pola atau data latih yang diinginkan dan kemudian menyimpannya ke dalam sebuah *array* 2 dimensi *trainInputs*[][]
43 - 47	Menampilkan data *array trainInputs* ke dalam bentuk tabel
48 - 56	Jika kondisi data tengah yang di ambil maka akan masuk logika ini yang akan memasukan data akhir ke dalam tabel

5.1.4 Set Bobot

Pada awal pelatihan nilai bobot ditentukan secara acak dengan rentang nilai antara 0 sampai dengan 1.

```
 1  private void set_bobot() {
 2      xawal.setRowCount(0);
 3      zawal.setRowCount(0);
 4      for(int j = 0 ; j < Hidden_Node ; j++)
 5      {
 6          bobot_w[j] = new Random().nextDouble() ;
 7          System.out.println("bobot w"+j+"1 = "+bobot_w[j]);
 8          for (int i = 0 ; i < Input_Node ; i++)
 9          {
10              if (j>0){
11              bobot_v[i][j] = new Random().nextDouble();
12              System.out.println("bobot v"+i+j+" = "
13              +bobot_v[i][j]);
14              }
15          }
16      }
17      zawal.addRow(new Object[]{bobot_w[0],bobot_w[1],
18      bobot_w[2],bobot_w[3],bobot_w[4],bobot_w[5],
19      bobot_w[6]});
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
20      for (int j = 1 ; j < 7 ; j++)
21      {
22          xawal.addRow(new Object[]{bobot_v[0][j],
23          bobot_v[1][j],bobot_v[2][j],bobot_v[3][j]
24          ,bobot_v[4][j]});
25      }
26  }
```

Kode Program 5.4 Inisialisasi Bobot

Penjelasan:

3 - 14	Menginisialisasikan bobot secara acak dan kemudian memasukan nilai tersebut kedalam sebuah *array bobot_w*[] dan *array bobot_v*[][]
17 - 23	Menampilkan nilai bobot yang sudah ditentukan ke dalam sebuah tabel

5.1.5 Proses Pelatihan

Proses pelatihan metode *Backpropagation* melalui tiga tahap yaitu tahap *feedforward, backpropagation* dan *weight update*. Proses pelatihan dilakukan untuk mendapatkan nilai bobot yang paling optimal.

```
 1  private void Pelatihan() {
 2      int pola_tf = Integer.parseInt(jml_pola.getText());
 3      double alpha = Double.parseDouble(jTextFieldAlpha
 4      .getText());
 5      int epoch = Integer.parseInt(jTextFieldIterasi
 6      .getText());
 7      iterasi = epoch;
 8      int simpaniterasi;
 9
10      double totaleror = 0;
11      tbl_mse.setRowCount(0);
12
13      for(int a = 0 ; a < iterasi ; a++)
14      {
15          for (int p = 0; p < pola_tf ; p++)
16          {
17                  //hitung_Z(p);
18                  for(int i = 0 ; i < 7 ; i++)
19                  {
20                      Znet[i] = 0.0;
21                      if (i>0){
```

```
22                      for(int j = 0; j < 5; j++)
23                      {
24                          Znet[i] += trainInputs[p][j]*
25                          bobot_v[j][i];
26                      }}
27                      Z[i] = 1 / (1+Math.exp(-Znet[i]));
28                      Z[0] = 1;
29                  }
30                  //hitung_Y(p);
31                  Ynet = 0.0;
32                  for(int i = 0 ; i < Hidden_Node ; i++)
33                  {
34                       Ynet += bobot_w[i] * Z[i];
35                  }
36                  Y[p] = 1 / (1+Math.exp(-Ynet));
37                  //hitung_errorY(p);
38                  nilai_error = (trainInputs[p][5]-Y[p])*
39                  Y[p]*( 1 - Y[p]);
40                  //hitung_deltaw(alpha);
41                  for(int i = 0 ; i < Hidden_Node ; i++)
42                  {
43                       delta_w[i] = alpha*nilai_error*Z[i];
44                  }
45                  //hitung_δnetj();
46                  for (int i = 1 ; i < Hidden_Node ; i++)
47                  {
48                       δ_netj[i] = nilai_error * delta_w[i];
49                  }
50                  // hitung δ_j();
51                  for (int i = 1 ; i < Hidden_Node ; i++)
52                  {
53                       δ_j[i] = δ_netj[i] * Z[i] *(1-Z[i]);
54                  }
55                  //hitung_selisihv(p, alpha);
56                  for (int i = 1 ; i < Hidden_Node ; i++)
57                  {
58                       for (int j = 0 ; j < Input_Node ; j++)
59                       {
60                            delta_vj[j][i] = alpha*δ_j[i]*
61                            trainInputs[p][j];
62                       }
63                  }
64
65                  //hitung_mseP(p,a);
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
66                  mseP[a] += Math.pow((Y[p]-
67                  trainInputs[p][5]), 2);
68                  if (p < pola_tf-1)
69                  {
70                      //update_bobot();
71                      for(int i = 0 ; i < Hidden_Node ; i++)
72                      {
73                          bobot_w[i] = bobot_w[i] + delta_w[i];
74                          for (int j=0 ; j<Input_Node ; j++)
75                          {
76                              if (i>0){
77                              bobot_v[j][i] = bobot_v[j][i]+
78                              delta_vj[j][i];
79                              }
80                          }
81
82                      }
83                  }
84          }
85      mseP[a] = mseP[a] / pola_tf;
86      totaleror = mseP[a];
87      if(a==0){bobot_baru();}
88      System.out.println("iterasi ke-"+a);if (a > 2){
89            if (mseP[a] > mseP[a-1])
90            {
91                  break;
92            }}
93      tbl_mse.addRow(new Object[]{a+1,mseP[a]});
94      }
95  }
```

Kode Program 5.5 Proses Pelatihan

Penjelasan:

10	Perulangan untuk menentukan jumlah maksimum iterasi
12	Perulangan untuk menentukan jumlah maksimum pola atau data latih
15 - 34	Perhitungan *feedforward* pada metode *backpropagation*
36 - 63	Perhitungan *backpropagation* pada metode *backpropagation*
67 - 83	Perhitungan *weight update* pada metode *backpropagation*
87 - 92	Jika nilai *error* sudah mencapai nilai yang sudah ditentukan maka proses pelatihan akan dihentikan

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

5.1.6 Proses Pengujian

Proses pengujian dilakukan untuk mengetahui tingkat akurasi dari hasil bobot yang didapatkan dari proses pelatihan. Pada proses pengujian fase *Backpropagation* hanya menggunakan tahap *feedforward* saja.

```
private void Pengujian() {
    int pola_tf = Integer.parseInt(jml_pola.getText());
    bobot_akhir();
    int tabel = Tabel_Data.getRowCount();
    double mse = 0;
    int a=1;
    for (int p = 0 ; p < pola_tf ; p++)
    {
        //hitung_Z(p);
        for(int i = 0 ; i < 7 ; i++)
        {
            Znet[i] = 0.0;
            if (i>0){
            for(int j = 0; j < 5; j++)
            {
                Znet[i] += trainInputs[p][j]*
                bobot_v[j][i];
            }}
            Z[i] = 1 / (1+Math.exp(-Znet[i]));
            Z[0] = 1;
        }
        //hitung_Y(p);
        Ynet = 0.0;
        for(int i = 0 ; i < Hidden_Node ; i++)
        {
            Ynet += bobot_w[i] * Z[i];
        }
            Y[p] = 1 / (1+Math.exp(-Ynet));
        mse += Math.pow((Y[p]-trainInputs[p][5]), 2);
    }
    mse = mse / pola_tf;
    ArrayList<Double> list = new ArrayList<Double>();
    for (int count = 0 ; count < tabel ; count++)
    {
        list.add(Double.parseDouble(Tabel_Data.
        getValueAt(count,4).toString()));
    }
    double max = Collections.max(list);
    double min = Collections.min(list);
```

```
40      int pola_tf = Integer.parseInt(jml_pola.getText());
41      double  ctual;
42      double prediksi;
43      dm_aktual.setRowCount(0);
44      dm_prediksi.setRowCount(0);
45      int b = Input_Node; //b=5
46      for (int I = 0 ; I < pola_tf ; i++)
47      {
48          double data = Double.parseDouble(Tabel_ExtData.
49          getValueAt(I, 5).toString());
50          ctual = ((max - min )*(data - 0.1)/0.8)+min;
51          prediksi = ((max - min )*(Y[i] - 0.1)/0.8)+min;
52          String aktString = String.format("%.1f",  ctual);
53          String predString = String.format("%.1f", prediksi);
54          int a = b - 1;//a=4
55          dm_aktual.addRow(new Object[]{Tabel_Data.
56          getValueAt(a,0),aktString});
57          dm_prediksi.addRow(new Object[]{Tabel_Data.
58          getValueAt(a,0),predString});
59          b++;
60      }
61
62      System.out.println("mse = " + mse);
63      String totalerorstring = String.format("%.12f", mse);
64      jTextFieldMSE.setText(totalerorstring);
65  }
```

Kode Program 5.6 Proses Pengujian

Penjelasan:

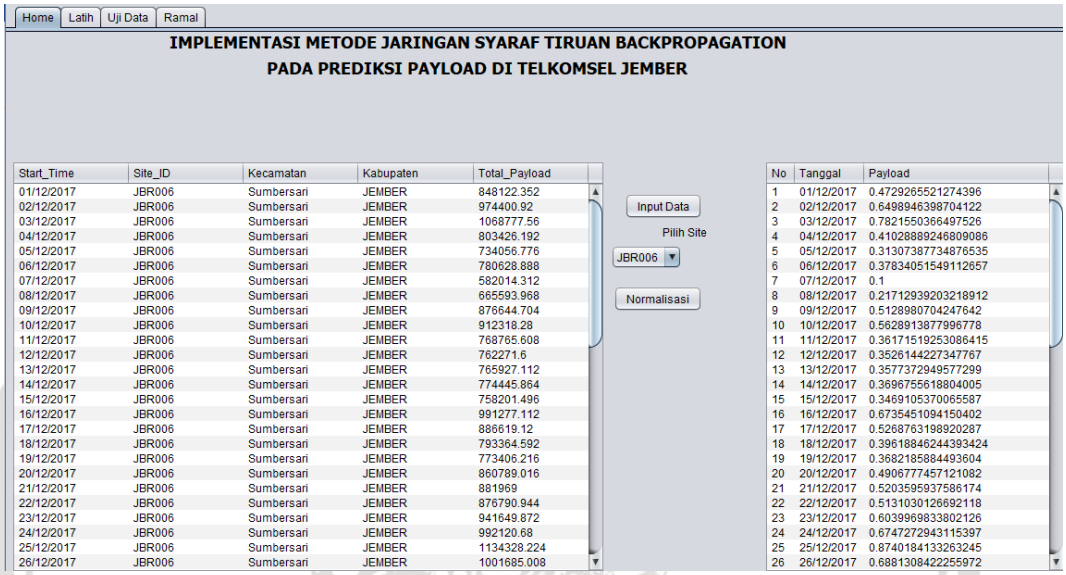
8 – 20	Perhitungan nilai *z* yang merupakan nilai *hidden layer* dengan menggunakan perulangan "For"
22 – 27	Perhitungan nilai *y* yang merupakan nilai *output layer* atau nilai prediksi yang dihasilkan dari proses pelatihan metode *backpropagation*
32 - 57	Proses perhitungan denormalisasi data yang hasilnya langsung ditampilkan pada sebuah tabel prediksi dengan tabel data aktual sehingga dapat dibandingkan antara data prediksi dengan data yang sebenarnya

5.2 Implementasi Tampilan Antarmuka

Pada subban tampilan antarmuka, akan dijelaskan serta diberika sebuah gambar tentang aplikasi prediksi *payload* 4G pada PT Telkomsel Jember dengan menerapkan metode jaringan saraf tiruan *backpropagation*.

5.2.1 Implementasi Halaman *Home*

Halaman *home* berisi fungsi *input* data, *filter* dan juga normalisasi. Untuk memperjelas penjelasan dapat dilihat pada Gambar 5.1.



Start_Time	Site_ID	Kecamatan	Kabupaten	Total_Payload
01/12/2017	JBR006	Sumbersari	JEMBER	848122.352
02/12/2017	JBR006	Sumbersari	JEMBER	974400.92
03/12/2017	JBR006	Sumbersari	JEMBER	1068777.56
04/12/2017	JBR006	Sumbersari	JEMBER	803426.192
05/12/2017	JBR006	Sumbersari	JEMBER	734056.776
06/12/2017	JBR006	Sumbersari	JEMBER	780628.888
07/12/2017	JBR006	Sumbersari	JEMBER	582014.312
08/12/2017	JBR006	Sumbersari	JEMBER	665593.968
09/12/2017	JBR006	Sumbersari	JEMBER	876644.704
10/12/2017	JBR006	Sumbersari	JEMBER	912318.28
11/12/2017	JBR006	Sumbersari	JEMBER	768765.608
12/12/2017	JBR006	Sumbersari	JEMBER	762271.6
13/12/2017	JBR006	Sumbersari	JEMBER	765927.112
14/12/2017	JBR006	Sumbersari	JEMBER	774445.864
15/12/2017	JBR006	Sumbersari	JEMBER	758201.496
16/12/2017	JBR006	Sumbersari	JEMBER	991277.112
17/12/2017	JBR006	Sumbersari	JEMBER	886619.12
18/12/2017	JBR006	Sumbersari	JEMBER	793364.592
19/12/2017	JBR006	Sumbersari	JEMBER	773406.216
20/12/2017	JBR006	Sumbersari	JEMBER	860789.016
21/12/2017	JBR006	Sumbersari	JEMBER	881969
22/12/2017	JBR006	Sumbersari	JEMBER	876790.944
23/12/2017	JBR006	Sumbersari	JEMBER	941649.872
24/12/2017	JBR006	Sumbersari	JEMBER	992120.68
25/12/2017	JBR006	Sumbersari	JEMBER	1134328.224
26/12/2017	JBR006	Sumbersari	JEMBER	1001685.008

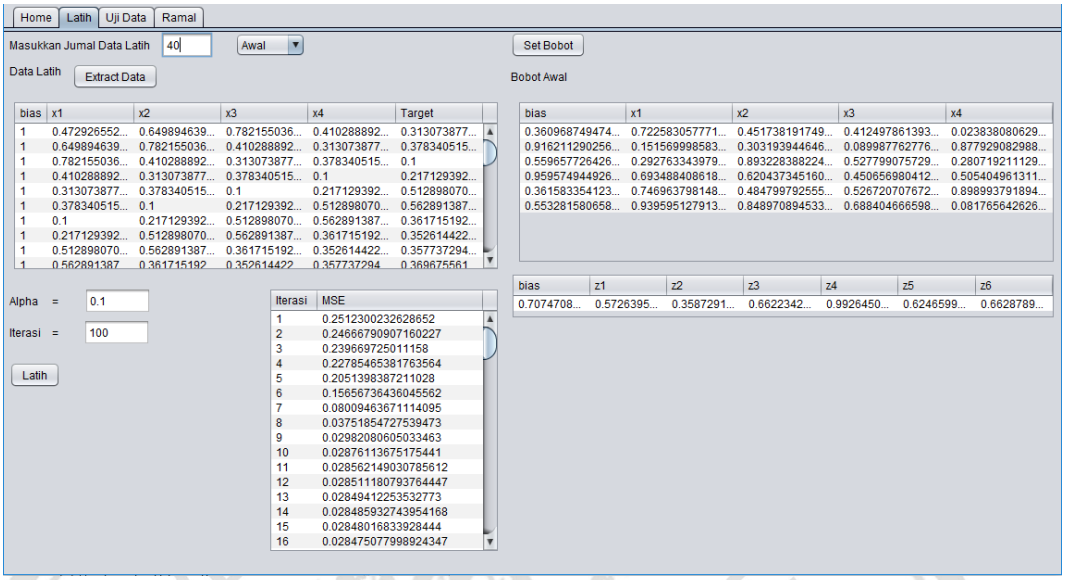
No	Tanggal	Payload
1	01/12/2017	0.4729265521274396
2	02/12/2017	0.6498946398704122
3	03/12/2017	0.7821550366497526
4	04/12/2017	0.41028889246809086
5	05/12/2017	0.31307387734876535
6	06/12/2017	0.37834051549112657
7	07/12/2017	0.1
8	08/12/2017	0.21712939203218912
9	09/12/2017	0.5128980704247642
10	10/12/2017	0.5628913877996778
11	11/12/2017	0.361715192530864l5
12	12/12/2017	0.3526144227347767
13	13/12/2017	0.3577372949577299
14	14/12/2017	0.3696755618804005
15	15/12/2017	0.3469105370065587
16	16/12/2017	0.6735451094150402
17	17/12/2017	0.5268763198920287
18	18/12/2017	0.39618846244393424
19	19/12/2017	0.3682185884493604
20	20/12/2017	0.4906777457121082
21	21/12/2017	0.5203595937586174
22	22/12/2017	0.5131030126692118
23	23/12/2017	0.6039969833802126
24	24/12/2017	0.6747272943115397
25	25/12/2017	0.8740184133263245
26	26/12/2017	0.6881308422255972

Gambar 5.1 Tampilan Halaman *Home*

Pada Gambar 5.1 terdapat 2 *button* yaitu *button input* data dan *button* normalisasi data serta satu *combo box* yang digunakan untuk mem-*filter* data yang ada di tabel data. Data yang terdapat di tabel, terdiri dari sekumpulan data *payload* 4G dari semua site yang berada di Kecamatan Sumbersari Jember. Agar dapat melakukan prediksi, maka perlu dilakukan *filter* data berdasarkan site ID. Karena peneliti ingin melakukan prediksi *payload* 4G pada tiap site. *Button input* data digunakan untuk memasukan data *payload* 4G yang memiliki ekstensi csv. Dan *button* normalisasi digunakan untuk menormalisasi data yang sudah di *filter* agar dapat dilakukan proses pelatihan.

5.2.2 Implementasi Halaman Latih

Pada halaman latih terdapat proses pelatihan metode *Backpropagation*. Untuk memperjelas penjelasan dapat dilihat pada Gambar 5.2.

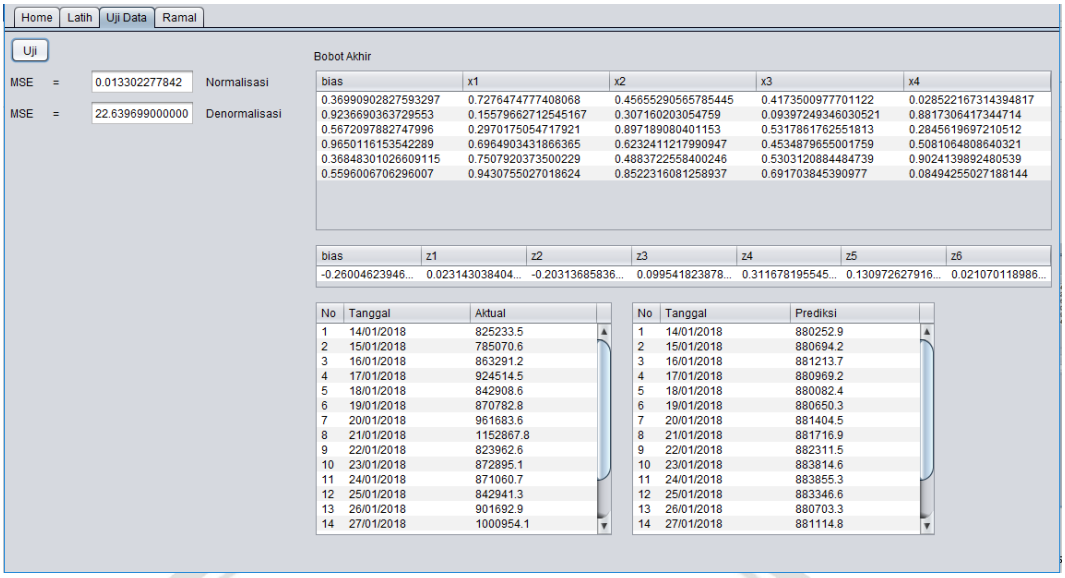


Gambar 5.2 Tampilan Halaman Latih

Gambar 5.2 telah menampilkan tampilan halaman latih yang memiliki beberapa *button* dan *text box*. *Button Extract* Data digunakan untuk menentukan jumlah data latih yang akan dilakukan pelatihan dengan cara memasukan jumlah data latih ke dalam *text box* yang berada di atasnya. Setelah mengklik *button Extract* Data maka aplikasi akan menampilkan data yang sudah tersusun dengan pola yang sudah ditentukan ke dalam sebuah tabel yang berada di bawahnya. *Button* kedua yaitu *button* set bobot yang berguna untuk menginisialisasikan bobot secara *random* dan ditampilkan ke dalam tabel bobot awal. *Button* terakhir yaitu *button* latih yang merupakan proses utama yang ada di halaman latih. *Button* latih memiliki proses pelatihan dengan menggunakan metode *Backpropagation* dengan menggunakan parameter–parameter yang sudah ditentukan sebelumnya. Sehingga aplikasi akan mendapatkan nilai MSE yang ditampilkan ke dalam sebuah tabel MSE.

5.2.3 Implementasi Halaman Uji

Halaman uji memiliki fungsi untuk menguji bobot yang sudah dihasilkan pada proses pelatihan metode *Backpropagation* pada proses sebelumnya. Untuk lebih jelasnya dapat dilihat pada Gambar 5.3.

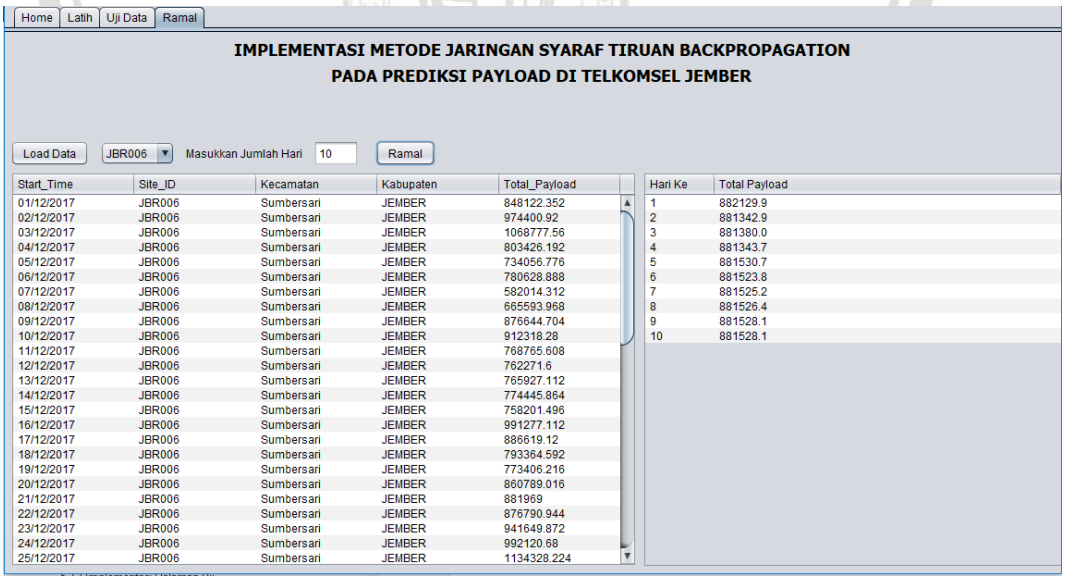


Gambar 5.3 Tampilan Halaman Uji

Pada Gambar 5.3 terlihat halaman uji yang hanya memiliki proses *feedforward*. *Button* uji akan memproses data latih yang sudah dinormalisasi sebelumnya dengan menggunakan bobot terakhir yang dihasilkan pada proses pelatihan dan akan menghasilkan sebuah data prediksi pada bulan selanjutnya yang ditampilkan pada tabel prediksi. Di sebelah tabel prediksi terdapat tabel aktual yang merupakan data yang sebenarnya, sehingga dapat digunakan untuk membandingkan data prediksi dengan data yang sebenarnya.

5.2.4 Implementasi Halaman Ramal

Halaman ramal digunakan untuk meramal data di bulan selanjutnya dari data yang dimasukan pada fungsi *load* data. Tampilan halaman ramal dapat dilihat pada Gambar 5.4.



Gambar 5.4 Tampilan Halaman Ramal

Button load data digunakan untuk memasukan data yang akan dilakukan prediksi. Data akan ditampilkan pada data ramal yang dapat dilihat pada Gambar 5.4. *Text box* masukan jumlah hari adalah jumlah data yang akan diprediksi oleh aplikasi. Setelah *Button* ramal diklik maka aplikasi akan memproses data yang ada dan akan menghasilkan data prediksi yang dapat dilihat pada tabel ramal yang berada pada sebelah kanan tabel data.

BAB 6 ANALISIS DAN PENGUJIAN

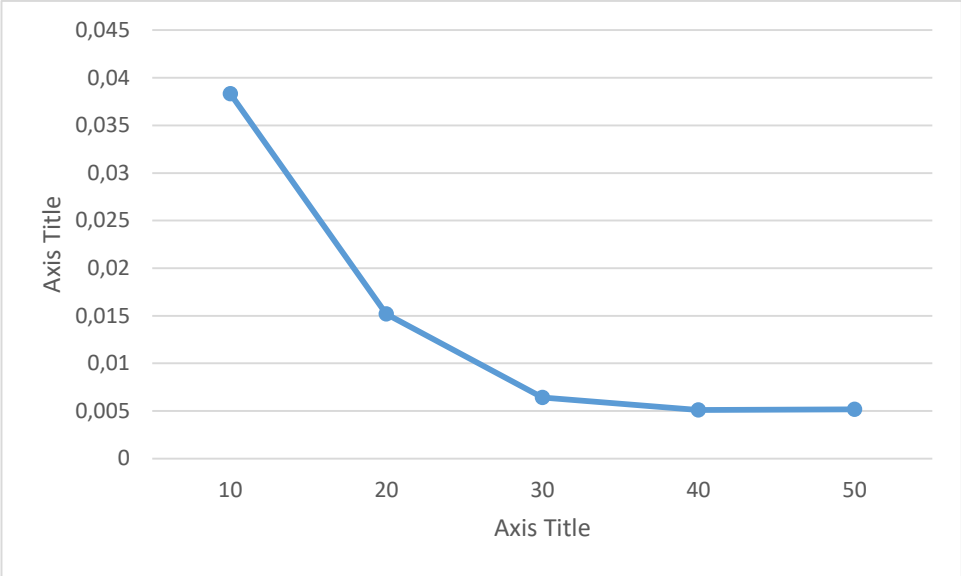
Pada bab ini dijelaskan tentang analisis dan pengujian dari hasil aplikasi peramalan *payload* 4G pada PT Telkomsel dengan menggunakan metode *Backpropagation*. Pengujian meliputi pengujian jumlah data, pengujian nilai *learning rate* dan pengujian jumlah iterasi maksimum. Pengujian dilakukan guna mengetahui jumlah data, nilai *learning rate* dan jumlah iterasi maksimum yang paling optimal, sehingga dapat memperoleh atau menghasilkan nilai *error* minimum.

6.1 Analisis dan Pengujian Jumlah Data Terhadap Nilai MSE

Pada pengujian jumlah data, jumlah data latih akan dilakukan pengujian jumlah data yang berbeda – beda dari 10 data latih sampai 50 data latih, sedangkan untuk data uji dilakukan pengujian sebanyak 10 data uji. Pengujian ini dilakukan menggunakan bobot akhir yang diperoleh dari pengujian data latih, kemudian dilakukan pengujian dengan menggunakan 10 data uji sehingga mendapatkan sebuah nilai MSE (*Mean Square Error*). Pengujian ini dilakukan sebanyak 5 kali yang kemudian dilakukan penghitungan rata–rata. Nilai *learning rate* yang digunakan adalah 0,1 dan jumlah iterasi maksimum 100, nilai ini sudah ditentukan oleh peneliti.

Tabel 6.1 Pengujian Jumlah Data Terhadap Nilai MSE

Latih / Uji	Percobaan Ke-					rata
	1	2	3	4	5	
10 10	0,0385	0,039824	0,036434	0,040865	0,036055	0,038335438
20 10	0,015453	0,014507	0,015214	0,01582	0,014992	0,015197089
30 10	0,006605	0,006157	0,006404	0,006504	0,006463	0,006426678
40 10	0,005097	0,005033	0,005181	0,005143	0,005087	0,005108067
50 10	0,005201	0,00499	0,005185	0,004978	0,005483	0,005167386



Gambar 6.1 Pengujian Jumlah Data

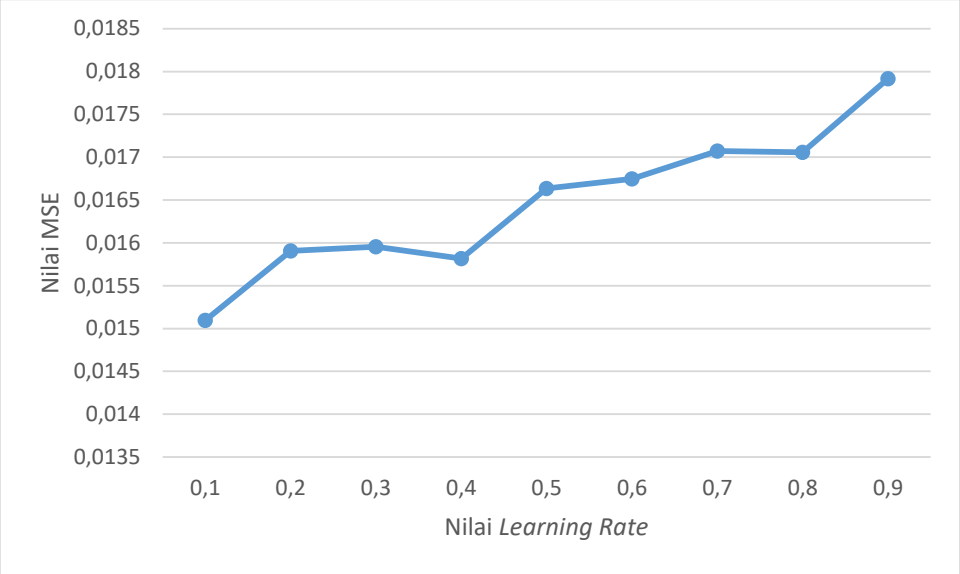
Dari Gambar 6.1 dapat dilihat bahwa jumlah data latih dari 10 sampai 40 data latih menghasilkan nilai MSE yang terus menurun, ini membuktikan bahwa jumlah data latih sebanyak 10 sampai dengan 40 data latih memiliki nilai MSE yang terus menurun. Namun pada jumlah data latih sebanyak 50 data latih, menghasilkan nilai *error* yang lebih besar dari jumlah data latih sebayak 40 data latih. Jadi jumlah data latih terbaik pada jumlah data latih sebanyak 40 data latih dan 10 data uji.

6.2 Analisis dan Pengujian Learning Rate Terhadap Nilai MSE

Pengujian *learning rate* dilakukan dengan menggunakan nilai *learning rate* antara 0,1 sampai dengan 0,9. Jumlah data latih yang digunakan sebanyak 10 data latih dan jumlah iterasi sebanyak 100 iterasi. Percobaan dilakukan sebanyak 5 kali yang pada akhirnya dilakukan perhitungan rata–rata. Sehingga dapat diketahui nilai *learning rate* yang memiliki nilai MSE yang terbaik.

Tabel 6.2. Pengujian Learning Rate Terhadap Nilai MSE

α	Percobaan Ke-					Rata -Rata
	1	2	3	4	5	
0,1	0.014625	0,015056	0,015582	0,014812	0,015401	0,015095095
0,2	0,01569	0,015965	0,016112	0,016067	0,015691	0,015905104
0,3	0,015731	0,015499	0,016064	0,016459	0,01603	0,015956785
0,4	0,015857	0,01595	0,015362	0,016203	0,015709	0,015816358
0,5	0,01709	0,016656	0,016636	0,016212	0,016578	0,016634542
0,6	0,017319	0,016798	0,016462	0,016894	0,016257	0,016745978
0,7	0,016331	0,017424	0,017175	0,016912	0,017527	0,017073676
0,8	0,016318	0,017325	0,01764	0,01637	0,017638	0,017058305
0,9	0,017994	0,017513	0,018209	0,017716	0,018152	0,017916575



Gambar 6.2. Pengujian *Learning Rate*

Gambar 6.2 menunjukkan nilai *learning rate* dari 0,1 sampai dengan 0,9 menghasilkan nilai MSE yang cenderung semakin besar, meskipun pada nilai *learning rate* 0,4 mengalami penurunan namun pada nilai *learning rate* 0,5 mengalami penaikan yang amat signifikan dan terus naik hingga nilai *learning rate* 0,9. Dan nilai *learning rate* yang memiliki nilai MSE terkecil adalah 0,1. Dalam kasus ini membuktikan bahwa besar nya nilai MSE dipengaruhi dengan besarnya nilai *learning rate*, jadi semakin besar nilai *learning rate* maka semakin besar pula nilai MSE yang dihasilkan.

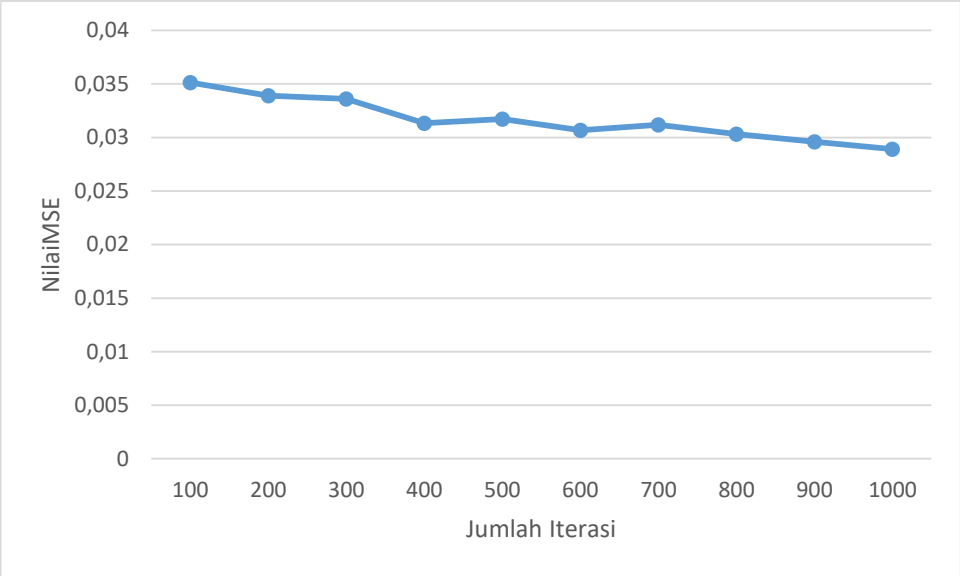
6.3 Analisis dan Pengujian Jumlah Iterasi Terhadap Nilai MSE

Pengujian iterasi dilakukan dengan menggunakan sebanyak 30 data latih tanpa menggunakan data uji dan nilai *learning rate* sebesar 0,1 dengan menggunakan nilai MSE yang dihasilkan dari proses pelatihan.

Tabel 6.3. Pengujian Jumlah Iterasi Terhadap Nilai MSE

iterasi	Percobaan Ke-					Rata -Rata
	1	2	3	4	5	
100	0,034539	0,034688	0,035295	0,035317	0,035835	0,035134734
200	0,033708	0,033101	0,034268	0,033924	0,034566	0,033913207
300	0,034859	0,032272	0,034281	0,032487	0,034119	0,033603718
400	0,030658	0,032286	0,032109	0,029702	0,031897	0,031330149
500	0,031286	0,030691	0,031324	0,03315	0,032173	0,031724832
600	0,030943	0,030075	0,031106	0,030918	0,030417	0,030692118
700	0,02975	0,030373	0,031518	0,031887	0,032417	0,03118916
800	0,030492	0,030919	0,032077	0,028709	0,029401	0,030319775

900	0,030499	0,030727	0,029372	0,028183	0,029299	0,029616032
1000	0,027944	0,028847	0,030263	0,028541	0,028967	0,028912482



Gambar 6.3 Pengujian Iterasi Terhadap Nilai MSE Latih

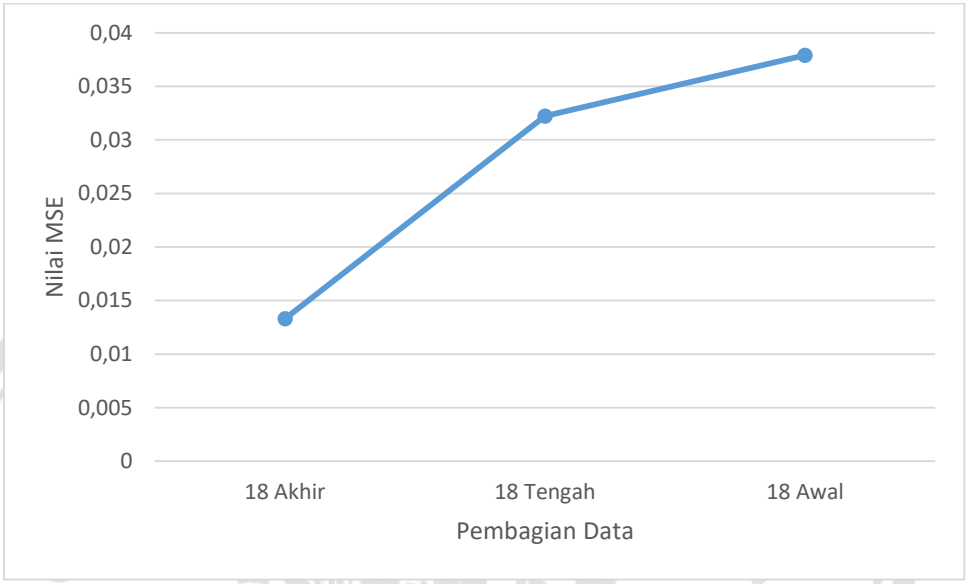
Dari Gambar 6.3 dapat dilihat bahwa nilai MSE Latih cenderung menurun dan banyaknya iterasi yang memiliki nilai MSE terkecil terletak pada iterasi ke 1000 karena pada proses pelatihan Metode *Backpropagation* nilai *error* yang dihasilkan diperbarui pada setiap iterasi sehingga semakin banyak iterasi yang dilakukan maka semakin kecil pula nilai *error* yang dihasilkan. Namun pada kasus tertentu bisa terjadi perbedaan.

6.4 Analisis dan Pengujian K-Fold Validation Terhadap Nilai MSE

Pengujian *K-Fold Validation* digunakan untuk mengetahui letak data latih dan data uji yang paling optimal dari sekumpulan data. Pengujian *K-Fold* Validation menggunakan 40 data latih dan 18 data uji yang diambil dari 58 data. Karena total data yang digunakan sebanyak 58 data, maka total data dibagi menjadi tiga bagian, yaitu akhir, tengah dan awal. Bagian akhir adalah 40 data latih awal yang terletak pada data ke-1 hingga data ke-40 dan sisanya sebagai data uji. Bagian tengah adalah 20 data awal sebagai data latih dan 18 data selanjutnya sebagai data uji sedangkan 20 data terakhir sebagai data latih. Dan bagian awal adalah data ke-1 hingga data ke-18 dijadikan sebagai data uji dan 40 data selanjutnya adalah data latih. Variable yang lain menggunakan nilai *learning rate* sebesar 0,1 dan jumlah iterasi maksimum sebanyak 100.

Tabel 6.4 Pengujian K-Fold Validation Terhadap Nilai MSE

Data Uji	Percobaan Ke-					Rata -Rata
	1	2	3	4	5	
18 Akhir	0,013348	0,013289	0,013336	0,01333	0,01324	0,013308481
18 Tengah	0,032392	0,033114	0,032622	0,030489	0,032449	0,032213293
18 Awal	0,036877	0,03868	0,03734	0,038711	0,037855	0,037892306

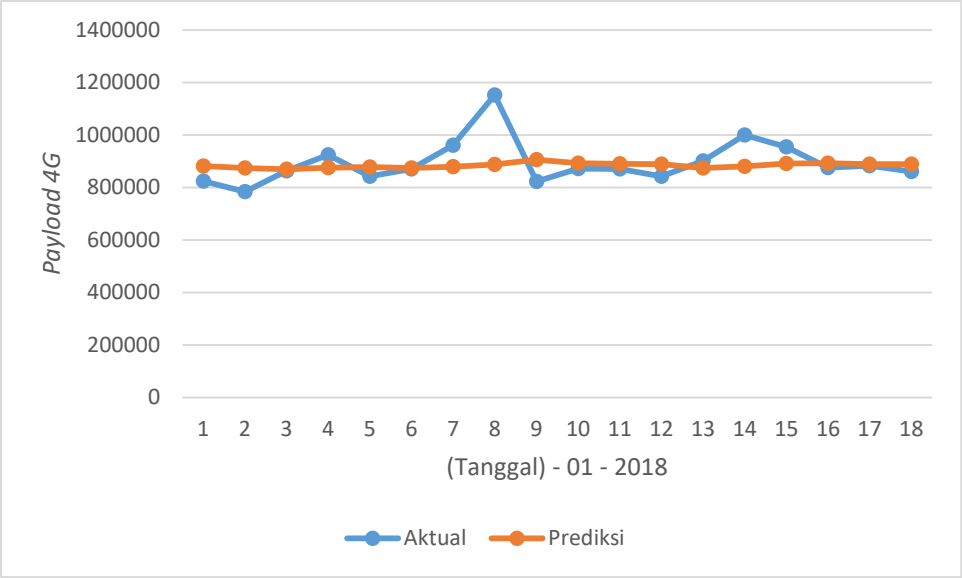


Gambar 6.4 Pengujian K-Fold Validation Terhadap Nilai MSE

Dari Gambar 6.4 dapat dilihat bahwa nilai MSE terbaik terletak pada bagian 18 akhir yaitu 18 data awal sebagai data uji dan 40 data selanjutnya sebagai data latih. Hal ini dapat membuktikan bahwa dari sekumpulan data yang terkumpul 18 data awal paling cocok untuk dijadikan sebagai data uji dan 40 data selanjutnya sebagai data latih terbukti dengan nilai MSE yang terkecil.

6.5 Analisis dan Pengujian Hasil Prediksi

Setelah melakukan pengujian nilai *learning rate,* jumlah iterasi maksimum, jumlah data dan pengujian *K-Fold Validation* maka didapatkan nilai–nilai variabel yang paling optimal untuk melakukan pengujian hasil dengan tujuan seberapa akurat metode *backpropagation* pada kasus prediksi *payload* 4G pada Telkomsel Jember. Tingkat akurasi akan dibuktikan dengan perhitungan nilai MAPE (*Mean Absolute Percentage Error*). Pengujian hasil penerapan metode *Backpropagation* pada prediksi *payload* 4G di Telkomsel Jember menggunakan nilai *learning rate* sebesar 0,1, iterasi maksimum sebanyak 1000 dan jumlah data latih sebanyak 40 yang terletak pada 40 data yang awal data 18 data uji yang terletak di akhir data.



Gambar 6.5 Pengujian Hasil

Dari Gambar 6.5 dapat dilihat bahwa pola dari hasil prediksi dengan menggunakan metode *Backpropagation* hampir mendekati pada data yang sebenarnya, meskipun pada tanggal 21 memiliki perbedaan antara data aktual dengan data prediksi yang cukup jauh yang memiliki selisih antara data actual dengan data prediksi yaitu sebesar 264088. Dengan menggunakan perhitungan MAPE (*Mean Absolute Percentage Error*) didapatkan nilai *error* sebesar 6,0154830745999%.

BAB 7 PENUTUP

Pada Bab Penutup ini diberikan sebuah kesimpulan dari penelitian dan hasil dari pengujian yang sudah dilakukan pada bab sebelumnya. Selain itu peneliti memberikan sebuah saran yang dapat dikembangkan oleh pembaca sehingga dapat dilakukan pada penelitian selanjutnya.

7.1 Kesimpulan

Setelah dilakukan beberapa pengujian, peneliti dapat mengambil sebuah kesimpulan yang dapat disebutkan sebagai berikut:

1. Nilai *learning rate* terbaik pada 0,1, jumlah iterasi terbaik pada 1000 iterasi dan data uji terbaik pada 18 data akhir dan 40 data awal. Hal ini dapat disimpulkan bahwa semakin besar nilai *learning rate* maka semakin besar pula nilai *error* yang dihasilkan, serta semakin banyak iterasi semakin baik pula nilai *error* yang dihasilkan. Namun pada kasus yang lain bisa saja terjadi perbedaan.
2. Dengan menggunakan nilai *learning rate* 0,1, iterasi maksimum sebanyak 1000 dan 18 data akhir sebagai data uji serta 40 data awal sebagai data latih didapatkan nilai MAPE sebesar 6,0154830745999%. hal ini membuktikan bahwa metode *Backpropagation* cukup baik untuk digunakan pada kasus prediksi *payload* 4G di Telkomsel Jember.

7.2 Saran

Saran yang dapat diberikan oleh peneliti untuk pembaca adalah dapat dilakukan penambahan optimasi agar hasil prediksi lebih baik dan akurat serta dapat memiliki nilai *error* yang lebih kecil. Algoritme optimasi yang dapat dipakai seperti algoritme genetika atau yang lainnya.

DAFTAR PUSTAKA

Amina, F. & Irawan, M. I., 2013. Prediksi Jumlah Penduduk Miskin Di Provinsi Kalimantan Selatan Menggunakan Jaringan Saraf Tiruan Backpropagation. *Seminar Nasional Pasca Sarjana XIII – ITS.*

Ariyanti, S., 201. Pemanfaatan Digital Divindend untuk layanan Long Term Evolution Area (LTE). *Puslitbang Sumber Daya dan perangkat Pos dan Informatika,* 11(3), pp. 189-208.

Diputra, M. I., 2018. Prediksi Tingkat Keuntungan Usaha Peternakan Itik Alabio Petelur menggunakan Jaringan Saraf Tiruan Backpropagation (Kasus di Kabupaten Hulu Sungai Utara Kalimantan Selatan). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(2), pp. 785-792.

Elfajar, A. B., 2017. Peramalan Jumlah Kunjungan Wisatawan Kota Batu Menggunakan Metode Time Invariant Fuzzy Time Series. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Februari, 1(2), pp. 85-94.

Haykin, S., 2009. *Nueral Networks and Learning Machines.* United State of America: Pearson.

Ika, R. C., 2018. Optimasi Fuzzy Time Series Menggunakan Algoritme Particle Swarm Optimization untuk Peramalan Nilai Pembayaran Penjaminan Kredit Macet. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(6), pp. 2364-2373.

Rachman, A. S., 2018. Peramalan Produksi Gula Menggunakan Metode Jaringan Saraf Tiruan Backpropagation Pada PG Candi Baru Sidoarjo. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* April, 2(4), pp. 1683-1689.

Raharyani, M. P., 2018. Implementasi Algoritme Support Vector Regression Pada Prediksi Jumlah Pengunjung Pariwisata. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* April, 2(4), pp. 1501-1509.

Riadi, M., 2016. *Jaringan Saraf Tiruan.* [Online] Available at: https://www.kajianpustaka.com/2016/11/jaringan-saraf-tiruan-jst.html

Rindhani, M. N., Putri, R. R. M. & Sutopo, 2017. Peramalan Dosis Pupuk Berdasarkan Karakteristik dan Lingkungan Tanaman Jeruk Siam Menggunakan Metode Backpropagation. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* November, 1(11), pp. 1214-1223.

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

Susanti, N., 2014. *PENERAPAN MODEL NEURAL NETWORK BACKPROPAGATION UNTUK PREDIKSI HARGA AYAM.* Jawa Tengah, s.n., pp. 325-332.

Taylor, J., 2003. Short-Term Electricity Demand Forecasting Using Double Seasonal Exponential Smoothing. *Journal of Operational Research Society,* pp. 799-805.

Telkomsel, 2017. *Sejarah Telkomsel.* [Online]
Available at: https://www.telkomsel.com/about-us/our-story/our-history

Wijaya, K. K., 2014. *Laporan Kinerja Telkomsel.* [Online]
Available at: https://id.techinasia.com/laporan-kinerja-finansial-xl-telkom-indosat-q3-2014

Yohanes, E., Mahmudy, W. F. & Rahmi, A., 2015. PENENTUAN UPAH MINIMUM KOTA BERDASARKAN TINGKAT INFLASI MENGGUNAKAN BACKPROPAGATION NEURAL NETWORK (BPNN). *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 04, 2(1), pp. 34-40.