OPTIMASI PERSEDIAAN MATERIAL TRANSFORMATOR MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN DAN *ANT COLONY OPTIMIZATION* DI PT. PLN (PERSERO) AREA JEMBER

SKRIPSI

oleh

Rizki Herdatullah

NIM. 122410101035

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2019

OPTIMASI PERSEDIAAN MATERIAL TRANSFORMATOR MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN DAN *ANT COLONY OPTIMIZATION* DI PT. PLN (PERSERO) AREA JEMBER

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Pendidikan Sarjana (S1) Program Studi Sistem Informasi dan mencapai gelar Sarjana Komputer

oleh

Rizki Herdatullah

NIM. 122410101035

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2019

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Allah SWT, yang telah memberikan kemudahan dan kelancaran dalam mengerjakan skripsi ini;
2. Almarhum Ayahanda Herman dan Ibunda Farida;
3. Saudara-saudaraku beserta keluarga besar;
4. Sahabatku beserta doa dan dukungannya;
5. Guru-guruku sejak taman kanak-kanak hingga perguruan tinggi;
6. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTTO

"Barang siapa yang menginginkan dunia maka hendaklah dengan ilmu, barang siapa yang menginginkan akhirat, maka hendaklah dengan ilmu, barang siapa yang menginginkan keduanya, maka hendaklah dengan ilmu". (HR. Tirmidzi)

PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama : Rizki Herdatullah

NIM : 122410101035

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Optimasi Persediaan Material Transformator Menggunakan Metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* di PT. PLN (Persero) Area Jember", adalah benar-benar hasil sendiri, adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, Juli 2019

Yang menyatakan

Rizki Herdatullah

NIM 122410101035

SKRIPSI

OPTIMASI PERSEDIAAN MATERIAL TRANSFORMATOR MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN DAN *ANT COLONY OPTIMIZATION* DI PT. PLN (PERSERO) AREA JEMBER

Oleh
Rizki Herdatullah
122410101035

PEMBIMBING

Dosen Pembimbing Utama : Prof. Dr. Saiful Bukhori, S.T., M.Kom
Dosen Pembimbing Pendamping : Windy Eka Yulia R, S. Kom, MT

PENGESAHAN PEMBIMBING

Skripsi berjudul ”Optimasi Persediaan Material Transformator Menggunakan Metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* di PT. PLN (Persero) Area Jember”, telah diuji dan disahkan pada,

hari, tanggal : Jumat, 12 Juli 2019

tempat : Program Studi Sistem Informasi Universitas Jember

Disetujui oleh:

Pembimbing I,	Pembimbing II,
Prof. Dr. Saiful Bukhori, S.T., M.Kom.	Windy Eka Yulia R, S. Kom, MT
NIP 196811131994121001	NIP 198403052010122002

PENGESAHAN PENGUJI

Skripsi berjudul ”Optimasi Persediaan Material Transformator Menggunakan Metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* di PT. PLN (Persero) Area Jember”, telah diuji dan disahkan pada,

hari, tanggal : Jumat, 12 Juli 2019

tempat : Program Studi Sistem Informasi Universitas Jember

Tim Penguji:

Penguji I,	Penguji II,
Oktalia Juwita, S. Kom., M.MT	Beny Prasetyo, S.Kom., M.Kom
NIP 198110202014042001	NRP 760016852

Mengesahkan

a.n Dekan

Wakil Dekan I Fakultas Ilmu Komputer,

Drs. Antonius Cahya P, M.App.Sc., Ph.D.

NIP. 196909281993021001

RINGKASAN

Optimasi Persediaan Material Transformator Menggunakan Metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* di PT. PLN (Persero) Area Jember; Rizki Herdatullah, 122410101035; 2019: 132 Halaman; Program Studi Sistem Informasi, Fakultas Ilmu Komputer Universitas Jember

PT PLN (Persero) Area Jember sebagai Badan Usaha Milik Negara yang ditugaskan oleh pemerintah untuk mengelola ketanagalistrikan berperan penting dalam menjaga ketersediaan pasokan dan pendistribusian tenaga listrik serta pelayanan produknya terhadap masyarakat Kabupaten Jember. Terkait dengan hal tersebut, terdapat dua indikator untuk mengukur kinerja PLN dalam menangani masalah keandalan pasokan dan pendistribusian tenaga listrik kepada pelanggan, yaitu SAIDI (*System Average Interruption Duration Index*) dan SAIFI (*System Interruption Frequency Index*).

Salah satu cara untuk mencapai kinerja SAIDI dan SAIFI yang baik adalah dengan melakukan optimasi ketersediaan material yang digunakan yakni salah satunya transformator atau biasa disebut trafo. Demi terwujudnya komitmen PLN dalam menurunkan nilai angka SAIDI dan SAIFI secara berkesinambungan, diperlukan manajemen optimasi material transformator yang lebih baik, sehingga ketika dibutuhkan *replacement*, material transformator yang dibutuhkan tersedia.

Optimasi berasal dari kata dasar optimal yang berarti terbaik, tertinggi, paling menguntungkan, menjadikan paling baik, dan perbuatan mengoptimalkan (menjadikan paling baik, paling tinggi, dan sebagainya). Salah satu cara yang dapat dilakukan adalah dengan melakukan prediksi. Prediksi adalah upaya untuk meramal masa depan. Prediksi dapat dilakukan dengan mempelajari pola data historis untuk menemukan suatu permodelan yang dapat menggambarkan data masa depan. Metode ini dinamakan prediksi *time series*. Salah satu algoritma yang dapat membentuk model dari data historis adalah *Artificial Neural Networks* (ANN). Algoritma ini meniru sistem saraf manusia sehingga dapat menyelesaikan masalah non-linear, seperti prediksi kebutuhan trafo dalam setahun.

Dalam proses permodelan, ANN akan selalu memperbarui bobot koneksinya sampai menemukan bobot yang optimum. Pada Tugas Akhir ini dibentuk sistem ANN yang dilatih oleh *Ant Colony Optimization* (ACO). Dari hasil pengujian dapat diketahui bahwa ANN dengan metode pembelajaran ACO dapat memprediksi kebutuhan trafo dengan hasil yang baik.

PRAKATA

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Karya Ilmiah Tertulis (Skripsi) berjudul "Optimasi Persediaan Material Transformator Menggunakan Metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* di PT. PLN (Persero) Area Jember".

Pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada :

1. Dr. Saiful Bukhori ST., M.Kom., Ketua Program Studi Sistem Informasi Universitas Jember sekaligus Dosen Pembimbing Utama, Windi Eka Yulia Retnani S.Kom.,MT, selaku Dosen Pembimbing Anggota, yang telah memberikan banyak arahan dan bimbingan dalam penulisan skripsi ini;
2. Alm. Ayah Herman dan Ibu Farida serta seluruh pihak keluarga yang telah memberikan dukungan dan doa yang tulus;
3. Akbar Anshari Hardikusumo sebagai sahabat yang telah menemani dan terus memberikan semangat kepada penulis selama penyelesaian skripsi ini.
4. Sahabat-sahabat terbaikku Program Studi Sistem Informasi angkatan 2012 FORMATION.
5. Semua pihak yang telah membantu baik tenaga maupun pikiran dalam pelaksanaan kegiatan penelitian dan penyusunan skripsi ini.
6. Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 12 Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1. PENDAHULUAN

Bab ini merupakan langkah awal dari penulisan tugas akhir. Bab ini berisi latar belakang, rumusan masalah, batasan masalah, metodologi penelitian, dan sistematika penulisan.

1.1. Latar Belakang

Setiap perusahaan memiliki tantangan untuk memperkirakan jumlah barang yang harus di produksi untuk memenuhi permintaan pasar yang terus berubah seiring waktu. Semakin baik perusahaan menangani tantangan ini, maka akan berdampak besar pada profitabilitas. Kelebihan jumlah pasokan akan berpengaruh pada ketersediaan tempat penyimpanan dan juga berpengaruh terhadap kualitas barang yang disimpan. Sebaliknya, apabila jumlah pasokan kurang dari pemintaan pasar maka perusahaan akan kehilangan kesempatan untuk mendapatkan keuntungan yang lebih besar.

Munculnya kondisi ketidakpastian dalam perusahaan untuk memenuhi permintaan pasar mendorong perusahaan untuk melakukan efisiensi dan optimasi dalam menjalankan proses bisnisnya. Efisiensi dan optimasi ini dapat dicapai dengan menerapkan manajemen terhadap persediaan secara tepat.

PT PLN (Persero) Area Jember sebagai Badan Usaha Milik Negara yang ditugaskan oleh pemerintah untuk mengelola ketanagalistrikan berperan penting dalam menjaga ketersediaan pasokan dan pendistribusian tenaga listrik serta pelayanan produknya terhadap masyarakat Kabupaten Jember. Terkait dengan hal tersebut, terdapat dua indikator untuk mengukur kinerja PLN dalam menangani masalah keandalan pasokan dan pendistribusian tenaga listrik kepada pelanggan, yaitu SAIDI (*System Average Interruption Duration Index*) dan SAIFI (*System Interruption Frequency Index*).

SAIDI adalah jumlah lama padam yang dirasakan rata-rata per pelanggan per tahun dan SAIFI adalah jumlah kali padam yang dirasakan rata–rata per pelanggan per tahun (Husna, Pelawi, & Yusniati, 2008). Buruknya penanganan terhadap kinerja dari SAIDI dan SAIFI dapat berpengaruh terhadap kepuasan pelanggan

yang berujung pada meningkatnya jumlah pengaduan, kerugian finansial berupa tuntutan ganti rugi, hilangnya Kwh jual, kerusakan aset dan sebagainya bagi pemasok listrik yang dalam hal ini adalah PT. PLN sendiri.

Salah satu cara untuk mencapai kinerja SAIDI dan SAIFI yang baik adalah dengan melakukan optimasi ketersediaan material yang digunakan yakni salah satunya transformator atau biasa disebut trafo. Demi terwujudnya komitmen PLN dalam menurunkan nilai angka SAIDI dan SAIFI secara berkesinambungan, diperlukan manajemen material transformator yang lebih baik, sehingga ketika dibutuhkan *replacement*, material transformator yang dibutuhkan tersedia.

Kebutuhan *replacement* yang tidak terprediksi, dapat disebabkan oleh banyak hal. Sebagai contoh, dapat dilihat pada kebutuhan *replacement* transformator. Berdasarkan informasi yang diperoleh dari pihak perusahaan, *replacement* transformator dilakukan akibat adanya kerusakan hubungan singkat jaringan, kegagalan proteksi, maupun bencana alam.

Artificial Neural Network (ANN) adalah suatu model komputasi yang mencoba untuk mensimulasikan jaringan syaraf biologis secara struktural dan/atau fungsional. ANN mengandung unit komputasi sederhana disebut neuron yang saling terhubung satu sama lain. ANN mampu mengidentifikasi dan mempelajari korelasi antara dataset input dan kecocokan nilai target. ANN dapat digunakan untuk melakukan prediksi, klasifikasi, dan asosiasi sebuah data (Bakirtzis & Petidis, 1996).

Ant Colony Optimization (ACO) adalah pendekatan metaheuristik dalam penyelesaian suatu masalah optimasi yang sulit. ACO sendiri terinspirasi dari perilaku semut untuk menemukan jalur terpendek antara dua tempat yakni sumber makanan dan sarang mereka yang memiliki jalur acak (Buana, 2014). Pada penelitian ini ACO digunakan saat ANN melakukan proses pembelajaran dengan cara menentukan nilai beban sesuai dengan model kerja feromon pada algoritma semut. Algoritma ACO yang digunakan dalam penelitian ini adalah *Continuous* ACO, karena proses pencarian bobot ANN akan menggunakan algoritma ACO dalam menemukan solusi ruang masalah yang bersifat berkelanjutan.

Pada penelitian sebelumnya algoritma ACO digunakan untuk melatih ANN melakukan prediksi nilai tukar rupiah dan dolar Amerika. Dalam penelitian lain digunakan untuk melakukan optimasi persediaan multi-item pada perusahaan retail. Pada penelitian ini akan melakukan optimasi terhadap persediaan trafo pada PT. PLN Area Jember agar dapat melakukan penanganan dengan tepat apabila terjadi gangguan listrik sehingga kinerja SAIDI dan SAIFI pada perusahaan tersebut dapat meningkat.

Berdasarkan penelitian di atas, dibutuhkan aplikasi berupa optimasi persediaan material transformator yang diharapkan dapat menjadi gambaran kepada perusahaan dalam menentukan jumlah material transformator yang harus disediakan pada wilayah pelayanan area jember pada tiap rayon di Kabupaten Jember.

1.2. Rumusan Masalah

Berdasarkan masalah yang telah diuraikan diatas, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menentukan jumlah persediaan transformator yang optimal tiap bulan di PT. PLN Area Jember menggunakan ANN dan ACO?
2. Bagaimana cara menguji efektifitas metode jaringan syaraf tiruan dan *ant colony optimization* sebagai metode pembelajarannya dalam memprediksi jumlah persediaan transformator yang harus disediakan pada tiap bulan?
3. Bagaimana mengimplimentasikan sistem optimasi persediaan transformator di PT. PLN Area Jember menggunakan metode jaringan syaraf tiruan dan *ant colony optimization* sebagai metode pembelajarannya ?

1.3. Tujuan

Tujuan dari penelitian yang dilakukan dalam skripsi ini adalah sebagai berikut:

1. Merumuskan optimasi persediaan transformator di PT. PLN Area Jember menggunakan metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* sebagai metode pembelajarannya.
2. Menguji efektifitas penerapan metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* sebagai metode pembelajarannya dalam memprediksi jumlah persediaan transformator yang harus disediakan pada tiap bulan.
3. Merancang dan membangun sistem informasi berbasis desktop untuk membantu PT. PLN Area jember dalam menentukan persediaan transformator tiap bulannya menggunakan metode Jaringan Syaraf Tiruan dan *Ant Colony Optimization* sebagai metode pembelajarannya.

1.4. Batasan Masalah

Pembahasan yang dilakukan dalam skripsi ini memiliki batasan masalah sebagai berikut:

1. Persediaan transformator hanya terbatas pada perencanaan jumlah transformator di bulan berikutnya.
2. Menggunakan data *history* jumlah persediaan transformator selama 5 tahun dari PT. PLN Area Jember.
3. Aplikasi yang dibangun adalah sistem informasi berbasis desktop.

1.5. Sistematika Penulisan

Sistematika penulisan dan urutan skripsi ini disusun sebagai berikut:

1. Pendahuluan

   Bab ini menjelaskan tentang latar belakang, perumusan masalah, tujuan dan manfaat, batasan masalah dan sistematika penulisan.
2. Tinjauan Pustaka

   Bab ini menjelaskan tentang materi, informasi, tinjauan pustaka, dan studi terdahulu yang menjadi kerangka pemikiran dalam penelitian, terutama penjelasan mengenai metode jaringan syaraf tiruan dan *ant colony optimization*.

3. Metodologi Penelitian

   Bab ini menjelaskan tentang metode penelitian yang digunakan, mulai dari jenis penelitian yang digunakan, tempat penelitian, teknik pengumpulan data, dan metode tahapan penelitian untuk pengembangan sistem.

4. Pengembangan Sistem

   Bab ini menjelaskan tentang gambaran umum pengembangan sistem, pengujian kinerja, pemeliharaan operasi sistem informasi.

5. Hasil dan Pembahasan

   Bab ini menjelaskan tentang hasil pengembangan dan pembahasan dari penelitian yang dilakukan.

6. Penutup

   Bab ini berisi tentang kesimpulan dari penelitian dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Pada bagian ini dipaparkan teori-teori dan pustaka yang akan dipakai dalam penelitian. Teori-teori ini berupa teori dari buku literatur dan jurnal. Berikut merupakan teori-teori yang dibahas dalam penelitian.

2.1. Penelitian Terdahulu

Pada penelitian sebelumnya optimasi menggunakan ANN dilakukan dengan cara mendapatkan jumlah pemesanan yang mengoptimumkan biaya total persediaan. Karena adanya fluktuasi permintaan yang dipengaruhi oleh hari raya Idul Fitri, maka penelitian tersebut menggunakan metode Variasi Kalender sebagai peramalan data permintaan. Hasil optimasi menunjukkan bahwa selisih biaya total persediaan perusahaan dan biaya total persediaan hasil Algoritma ANN, sangat kecil dan tidak signifikan (Wulandari, Mumpuni, & Irhamah, 2011).

Pada penelitian yang lain salah satu metode yang dapat digunakan dalam melakukan prediksi adalah menggunakan metode *time series*, yakni prediksi yang dihasilkan berdasarkan data *history*. Sama halnya dengan trafo yang merupakan data non-linear dengan banyak *noise*, hal ini dapat mempersulit ANN dalam membaca pola data. Solusi yang dipakai adalah dengan menggunakan algoritma ACO sebagai metode pembelajarannya. Dari hasil pengujian dapat diketahui bahwa ANN dengan metode pembelajaran ACO dapat memprediksi nilai tukar IDR/USD dengan hasil yang baik dilihat dari nilai *Mean Absolute Error* (MAE) hasil prediksi yang cukup kecil. (Kho, Suyanto, & Baizal, 2011).

Penelitian ini dilakukan berdasarkan teori dan studi yang pernah dilakukan sebelumnya, maka penulis melakukan optimasi ACO yang akan digunakan saat ANN melakukan proses pembelajaran dengan cara menentukan nilai beban sesuai dengan model kerja feromon pada algoritma semut. Algoritma ACO yang digunakan dalam penelitian ini adalah *Continuous* ACO, karena proses pencarian bobot ANN akan menggunakan algoritma ACO dalam menemukan solusi ruang masalah yang bersifat berkelanjutan.

2.2. Optimasi

Optimasi berasal dari kata dasar optimal yang berarti terbaik, tertinggi, paling menguntungkan, menjadikan paling baik, dan perbuatan mengoptimalkan (menjadikan paling baik, paling tinggi, dan sebagainya). Sedangkan secara istilah optimasi adalah proses untuk mencapai hasil ideal atau optimal (nilai efektif yang dapat dicapai) (Departemen Pendidikan Indonesia, 2008).

Dalam penelitian ini optimasi merupakan proses untuk mengoptimalkan persediaan transformator terhadap kebutuhan yang akan datang sehingga pada saat terjadi gangguan listrik perusahaan tidak kekurangan material transformator.

2.3. Transformator

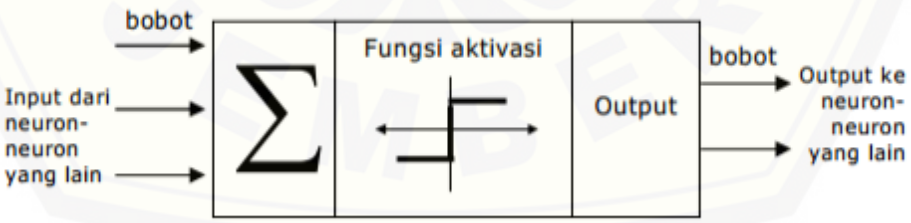
Transformator merupakan peralatan listrik yang berfungsi untuk menyalurkan daya/tenaga dari tegangan tinggi ke tegangan rendah atau sebaliknya. Transformator menggunakan prinsip hukum induksi faraday dan hukum lorentz dalam menyalurkan daya, dimana arus bolak balik yang mengalir mengelilingi suatu inti besi maka inti besi itu akan berubah menjadi magnet. Transformer atau trafo merupakan suatu peralatan yang dapat mengubah tenaga listrik dari suatu level tegangan ke level tegangan lainnya (Wibowo, 2018).

Trafo biasanya terdiri atas dua bagian inti besi atau lebih yang dibungkus oleh belitan-belitan kawat tembaga. Prinsip pengubahan level tegangan dilakukan dengan memanfaatkan banyaknya jumlah belitan pada inti trafo. Bila salah satu kumpulan belitan, biasanya disebut belitan primer (N1), diberikan suatu tegangan yang berubah-ubah, maka akan menghasilkan *mutual flux* yang berubah-ubah dengan besar *amplitude* yang tergantung pada tegangan, frekuensi tegangan, dan jumlah lilitan kawat tembaga di belitan primer. *Mutual flux* yang terjadi akan terhubung dengan belitan lain yang disebut sisi sekunder (N2) dan akan menginduksi suatu tegangan yang berubah-ubah di dalamnya dengan nilai tegangan yang bergantung pada jumlah lilitan pada belitan sekunder. Dengan mengatur perbandingan jumlah lilitan antara sisi primer dan sekunder, maka akan dapat ditentukan rasio tegangan ataupun sering disebut rasio trafo.

2.4. Jaringan Syaraf Tiruan

Jaringan syaraf adalah suatu arsitektur jaringan untuk memodelkan kerja dari sistem syaraf manusia yaitu otak di dalam melaksanakan tugas tertentu (Suyanto, 2008). Pemodelan tersebut didasarkan pada kemampuan otak manusia dalam mengorganisir sel-sel penyusunnya yang dikenal dengan neuron, sehingga memiliki kemampuan untuk melaksanakan tugas-tugas tertentu seperti pengenalan pola dengan efektivitas jaringan yang sangat tinggi.

Jaringan saraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut (Hermawan, 2006). Istilah buatan di sini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. Seperti halnya otak manusia, jaringan syaraf juga terdiri dari beberapa neuron, dan ada hubungan antara neuron-neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada jaringan syaraf, hubungan ini dikenal dengan nama bobot. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. Menurut Kusumadewi, dkk (2006) model struktur untuk jaringan saraf tiruan dapat dilihat pada gambar 2.1



Gambar 2. 1 Model Struktur Jaringan Saraf Tiruan (Kusumadewi, Hartati, & Harjoko, 2006)

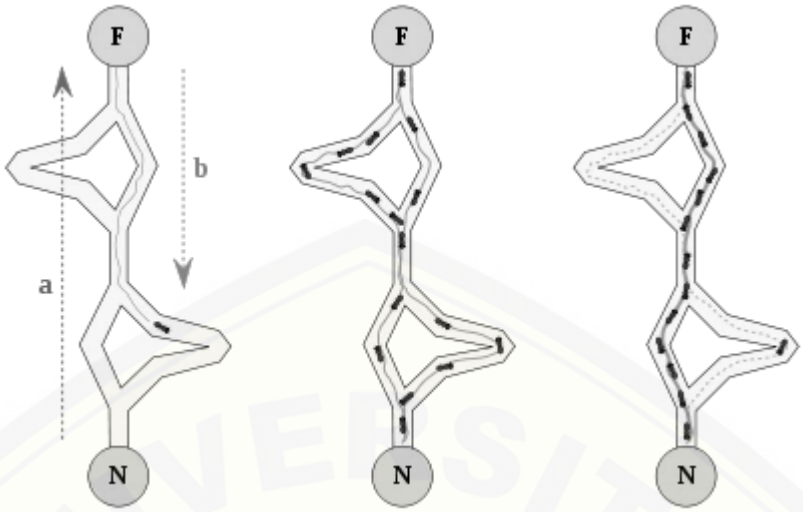
Informasi akan dikirim ke neuron dengan bobot tertentu, input ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang diinputkan sehingga hasil penjumlahan tersebut akan dibandingkan

dengan nilai ambang (*threshold*) tertentu melalui suatu proses yang disebut dengan proses aktivasi.

Pada jaringan saraf neuron-neuron dikumpulkan kedalam sebuah lapisan yang disebut dengan lapisan neuron (*neuron layers*), semua neuron dalam sebuah lapisan akan dihubungkan dengan lapisan lainnya baik itu sesudah dan sebelumnya kecuali dibagian input dan ouput yang dikenal dengan lapisan tersembunyi (*hidden layers*).

2.5. *Ant Colony Optimization*

Ant Colony Optimization merupakan algoritma yang terinspirasi pada perilaku semut dalam menentukan jalur terpendek dari sarang dan sumber makanan. Awalnya semut secara acak melakukan perjalanan mencari makanan tanpa adanya jalur. Pada saat semut melakukan perjalanan ke sumber makanan dan kembali ke sarang, semut melepaskan senyawa bernama feromon. Feromon memiliki sifat yakni akan menguap seiring waktu dan semut akan mengikuti jejak feromon yang memiliki konsentrasi terkuat. Seiring waktu akan semakin banyak semut yang mengikuti jejak yang memiliki konsentrasi feromon tertinggi dan akhirnya akan membentuk jalur terpendek karena endapan feromon pada jejak yang panjang akan lebih cepat menguap daripada jejak yang lebih pendek disebabkan semut membutuhkan waktu relatif lebih lama untuk melintasi lintasan yang panjang (Greene & Gordon, 2007).



Gambar 2. 2 Perjalanan Semut Menuju Sumber Makanan dan Kembali Ke Sarangnya (Greene & Gordon, 2007)

Terdapat banyak variasi ACO dalam menyelesaikan suatu masalah pencarian solusi, salah satunya *Continuous* ACO yang akan digunakan pada penelitian ini. *Continuous* ACO digunakan untuk mencari solusi dalam suatu masalah optimasi yang bersifat berkelanjutan (Hu, Zhang, & Li, 2008).

2.5.1. Algoritma *Continuous Ant Colony Optimization*

Terdapat beberapa komponen pada algoritma *Continuous Ant Colony Optimization* yakni sebagai berikut:

1. Komponen Arsip

Komponen arsip bertujuan mengatur arsip solusi dimana kandidat solusi tersimpan; setiap kandidat solusi mengandung nilai untuk setiap variabel ‘n’ yang menunjukkan ruang pencarian. Dalam kasus ini, ‘n’ adalah berat jaringan neuron dan ‘k’ merupakan arsip solusi.

2. Komponen *Fitness Vector*

Fitness Vector merupakan fungsi objektif $f(S_1)$ untuk semua solusi dalam arsip. Dalam kasus ini fungsi mencoba meminimalkan kesalahan jumlah kuadrat untuk set pelatihan. Arsip diurutkan berdasarkan nilai dari *Fitness Vector*.

3. Komponen *Weight Vector*

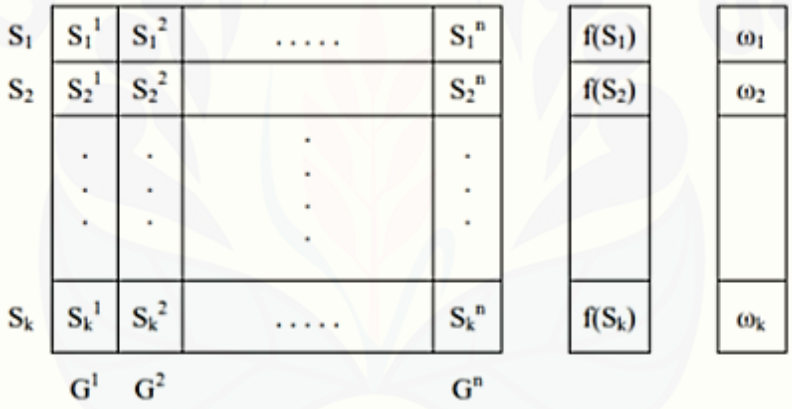
Vektor ini mengandung berat yang menentukan setiap solusi, dengan kalkulasi formula sebagai berikut:

$$\omega_l = \frac{1}{q_k \sqrt{2\pi}} e^{\frac{(\ell-1)^2}{2q^2k^2}} \qquad \text{persamaan (2.1)}$$

Dimana, jika parameter q kecil maka solusi berkualitas tinggi akan dipilih dan jika besar maka distribusi bobotnya hampir sama (l adalah indeks solusi dalam arsip).

4. Komponen *Neural Network*

Jaringan *neural multilayer feed-forward* yang digunakan untuk pengembangan ini dijelaskan pada gambar 2.3.



Gambar 2. 3 Struktur Arsip, *Fitness Vector*, dan *Weight Vector* (Pandian, 2013)

2.5.2. Pencarian Solusi

Proses pencarian solusi pada algoritma *Continuous Ant Colony Optimization* yakni sebagai berikut:

1. Inisialisasi Arsip

Arsip diinisialisasi dengan solusi acak k dengan nilai setiap berat dalam kisaran [-1,1].

2. *Probability Density Function* (PDF)

Untuk konstruksi probabilistik solusi, digunakan *Gaussian PDF* untuk menentukan distribusi probabilitas di ruang pencarian. Distribusi *Gaussian* adalah distribusi berbentuk lonceng yang berpusat pada *mean*. Untuk membuat distribusi *Gaussian* dibutuhkan dua parameter μ (*mean*) dan σ (standar deviasi).

3. *Gaussian Kernel* PDF

Fungsi *Gaussian* memiliki kelemahan dikarenakan variasi dalam bentuk fungsi *Gaussian* terbatas, fungsi *Gaussian* tunggal tidak dapat digunakan untuk menentukan situasi di mana ada dua area terpisah. Jadi, pada proyek ini digunakan *Gaussian Kernel PDF* yang disempurnakan dimana merupakan jumlah tertimbang dari beberapa fungsi *Gaussian* satu dimensi g^i_l (x) seperti yang tercantum pada persamaan berikut:

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i\,2}}}$$

persamaan (2.2)
(Pandian, 2013)

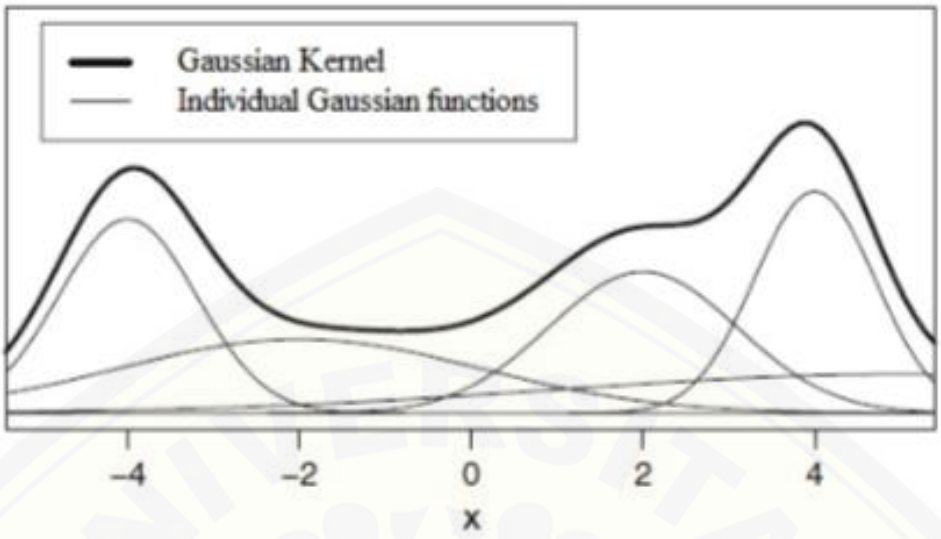
dimana:

k = jumlah dari solusi didalam arsip,

l = indeks dari solusi, dan

i = dimensi.

Kernel PDF membantu kita untuk membuat ruang distribusi probabilitas n dimensi seperti yang ditunjukkan pada Gambar 2.4 yang dapat disampel untuk menghasilkan seperangkat bobot yang bias untuk jaringan saraf yang sedang dilatih.



Gambar 2. 4 Gambaran dari *Gaussian Kernel* PDF
(Pandian, 2013, *Training Neural Networks With Ant Colony Optimization* Hal. 11)

4. Sampling Gaussian Kernel PDF

Masing-masing dari setiap dimensi dari solusi memiliki *Gaussian Kernel* PDF – G (x), dibangun menggunakan beban yang sama dengan dimensi yang ada didalam arsip. Untuk menemukan nilai dimensi *i* (beban NN) dari solusi yang baru terbentuk, maka $G^i(x)$ menjadi sampel. Sampling dilakukan dalam dua fase.

a. Salah satu fungsi *Gaussian* yang menyusun *Gaussian Kernel* PDF dipilih dengan probabilitas memilih fungsi *Gaussian* ke-l yang diberikan di bawah ini

$$\rho_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \qquad \text{persamaan (2.3)}$$

(Pandian, 2013)

b. Fungsi *Gaussian* yang dipilih, disampel untuk mendapatkan nilai untuk dimensi. Setelah fungsi *Gaussian* dipilih, fungsi yang sama (yaitu indeks l dari arsip) digunakan untuk menghasilkan nilai untuk semua dimensi dari solusi itu, setiap kali dengan memasukkan nilai-nilai untuk rata-rata μ_l dan σ_l dari setiap *i* dari solusi *l* di arsip.

Rata-rata setara dengan nilai dari variabel, contoh *i* variabel dari semua solusi didalam arsip menjadi elemen vektor μ_i.

Standar deviasi dihitung sebagai rata-rata jarak antara variabel terpilih dari solusi terpilih dengan variabel yang sama didalam semua solusi-solusi yang ada di arsip.

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{x_e^i - x_l^i}{k-1}$$ persamaan (2.4) (Pandian, 2013)

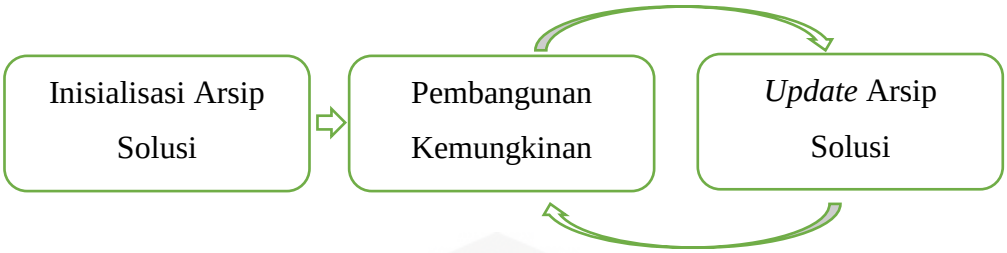
Dimana standar deviasi dikalikan dengan ξ, yang beranalog dengan laju penguapan feromon di ACO untuk optimasi kombinatorial. Tinggi nilai dari ξ untuk hasil dalam kecepatan konvergensi rendah.

Dengan demikian, semua *n* Gaussian Kernel PDF diambil sampelnya untuk membuat solusi. Pengambilan sampel ini adalah proses yang diulang untuk membuat solusi *k* (set bobot jaringan saraf) untuk setiap iterasi pelatihan

2.6. Jaringan Syaraf Tiruan dan *Ant Colony Optimization*

Algoritma yang digunakan dalam sistem ini adalah termasuk kategori *Continuous Ant Colony Optimization*, sebagai solusi masalah ruang pencarian serangkaian "beban" jaringan saraf tiruan yang terus-menerus dioptimalkan. Jadi, dalam penelitian ini *Ant Colony Optimization* akan terus-menerus melakukan optimasi terhadap Jaringan Syaraf Tiruan sebagaimana didefinisikan oleh Krzysztof Socha dan Christian Blum. Berikut penjelasan dari algoritma *Continuous Ant Colony Optimization* beserta ilustrasi yang ditunjukkan pada Gambar 2.3 (Pandian, 2013):

1. Membangun solusi kandidat probabilistik beban menggunakan distribusi probabilitas atas ruang pencarian.
2. Gunakan kandidat solusi untuk memodifikasi distribusi probabilitas dengan cara pembentukan sampel beban bias terhadap solusi yang memiliki kualitas tinggi.



Gambar 2. 5 Diagram Alir Jaringan Syaraf Tiruan dengan metode pembelajaran *Ant Colony Optimization* (Pandian, 2013)

Untuk masalah optimasi, algoritma ACO menggunakan model feromon untuk secara probabilistik membangun solusi, jejak feromon bertindak sebagai memori yang menyimpan pengalaman algoritma pencarian. Dalam ACO untuk optimasi berkelanjutan, pencarian dimodelkan dengan n variabel, yang merupakan jumlah bobot jaringan saraf dalam kasus kami. Kandidat solusi disimpan dalam arsip dan digunakan untuk mengubah distribusi probabilitas di ruang solusi. Distribusi probabilitas analog dengan model feromon yang digunakan dalam masalah optimasi.

2.7. Metode Pengukuran Tingkat *Error* Hasil Prediksi

Membandingkan kesalahan peramalan adalah suatu cara sederhana, apakah suatu teknik peramalan tersebut patut dipilih untuk digunakan membuat peramalan data yang sedang kita analisa atau tidak. Minimal prosedur ini dapat digunakan sebagai indikator apakah suatu teknik peramalan cocok digunakan atau tidak. Dan teknik yang mempunyai MSE *(Mean Squared Error)* terkecil merupakan ramalan yang terbaik (Nachrowi & Hardius, 2006).

Menurut Vincent Gaspers (1998, hal. 80) dalam bukunya menyebutkan akurasi peramalan akan semakin tinggi apabila nilai-nilai MAD, MSE, dan MAPE semakin kecil. Cara yang cukup sering digunakan dalam mengevaluasi hasil peramalan yaitu dengan menggunakan metode *mean absolute percentage error*. Suatu model mempunyai kinerja sangat bagus jika nilai MAPE berada dibawah 10% dan mempunyai kinerja bagus jika nilai MAPE berada diantara 10% dan 20%. Tiga ukuran berikut sering digunakan:

1. *Mean Absolute Deviation*

Metode untuk mengevaluasi metode peramalan menggunakan jumlah dari kesalahan-kesalahan yang absolut. *Mean Absolute Deviation* (MAD) mengukur ketepatan ramalan dengan merata-rata kesalahan dugaan (nilai absolut masing-masing kesalahan). Nilai MAD dapat dihitung dengan menggunakan persamaan (2.5).

$$MAD = \frac{\sum |X_t - F_t|}{t}$$ persamaan (2.5) (Kader, 1999)

dimana:

t = jumlah periode

X_t = permintaan pada periode t

F_t = ramalan untun periode t

2. Kesalahan Rata-Rata Kuardrat *(Mean Square Error)*

Mean Square Error (MSE) merupakan cara kedua untuk mengukur kesalahan peramalan keseluruhan. MSE merupakan rata-rata selisih kuadrat antara nilai yang diramalkan dan yang diamati. Kekurangan penggunaan MSE adalah bahwa ia cenderung menonjolkan deviasi yang besar karena adanya pengkuadratan. Rumus untuk menghitung MSE ditunjukkan pada persamaan (2.6).

$\text{MSE} = \frac{\sum_{t=0}^{n}(X_t - F_t)^2}{n}$;untuk deviasi rata-rata kuadrat persamaan (2.6) (Kader, 1999)

dimana:

X_t = permintaan pada periode t

F_t = ramalan untuk periode t

n = total jumlah periode

3. Persentase kesalahan absolut rata-rata (*Mean Absolute Percentage Error* – MAPE)

MAPE merupakan persentase yang dihitung dari nilai absolut kesalahan di masing-masing periode dan dibagi dengan jumlah data aktual periode tersebut kemudian dicari rata-rata kesalahannya. Rumus untuk menghitung MAPE ditunjukkan pada persamaan (2.7).

persamaan (2.7)

$$MAPE = \frac{\frac{\sum|X_t - F_t|}{X_t} x100\%}{n}$$; *deviasi* presentasi absolut (de Myttenaere, 2016)

dimana:

X_t = permintaan pada periode t

F_t = ramalan untuk periode t

n = total jumlah periode

|| = nilai absolut

BAB 3. METODE PENELITIAN

Bab ini merupakan penguraian jenis penelitian, tempat dan waktu penelitian, rancangan penelitian, pengumpulan data, dan tahapan penelitian.

3.1. Jenis Penelitian

Jenis penelitian yang dilakukan adalah penelitian kualitatif dan kuantitatif. Penelitian kualitatif dalam penelitian ini dilakukan dengan cara melakukan wawancara kepada PT. PLN Area Jember selaku pengambil keputusan mengenai bagaimana tata kelola persediaan trafo dan kebutuhan tiap bulannya serta melakukan pengumpulan data *history* selama 5 tahun terakhir.

Sedangkan metode kuantitatif yang digunakan adalah pada tahap pengumpulan data dan analisis dalam bentuk angka serta penelitian ini juga mengkaji teori dan metode yang sudah ada sebelumnya.

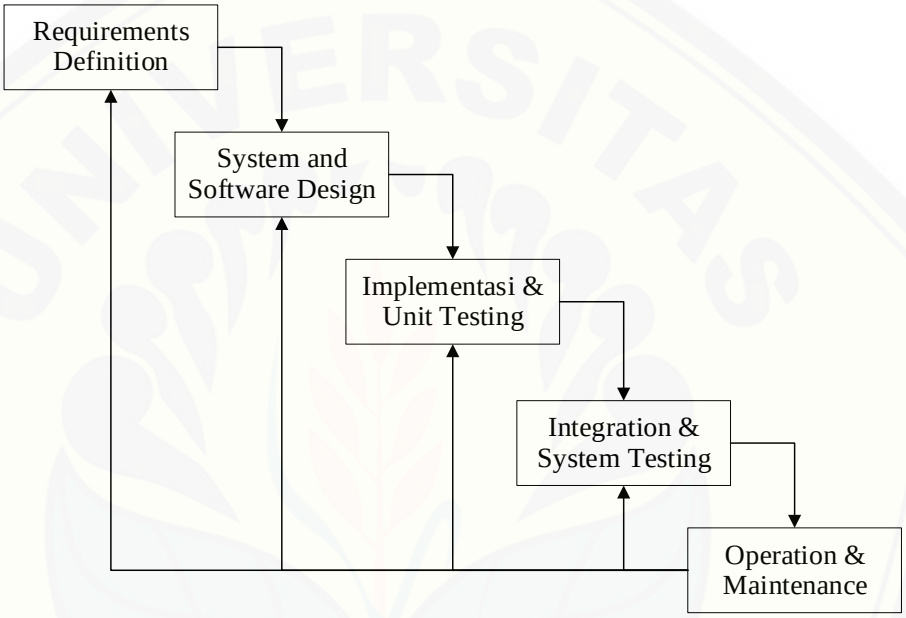
3.2. Tempat dan Waktu Penelitian

Penelitian dilakukan di PT. PLN Area Jember yang beralamatkan Jl. Gajah Mada No.198, Kaliwates, Kec. Kaliwates, Kab. Jember, Jawa Timur. Waktu penelitian dilaksanakan selama 3 (tiga) bulan yaitu pada bulan Oktober 2016 sampai dengan Desember 2016. Tahap penelitian selanjutnya dilakukan di Kampus Program Studi Sistem Informasi Universitas Jember beralamatkan Jalan Kalimantan 37 Kab. Jember dengan waktu penelitian dilakukan pada bulan Januari 2016 hingga Januari 2017.

3.3. Tahapan Penelitian

Proses awal yang dilakukan pada penelitian ini adalah pengumpulan data, studi literatur, dan/atau studi lapangan. Selanjutnya, penelitian ini melakukan perumusan masalah yang dilanjutkan perumusan tujuan dan manfaat. Proses-proses tersebut merupakan tahapan analisis dan definisi persyaratan pada model *waterfall*. Proses selanjutnya adalah implementasi sistem yang merupakan tahapan

implementasi dan pengujian unit pada model *waterfall*. Kemudian, dilakukan proses pengujian dan evaluasi sistem yang merupakan tahapan integritas dan pengujian sistem yang dilanjutkan ke tahapan operasi dan pemeliharaan pada model *waterfall*. Setelah itu, dilakukan proses analisis deskriptif berupa hasil dan pembahasan sebagaimana yang dipaparkan pada bab 5 hingga rancangan penelitian ini berakhir pada proses penarikan kesimpulan dan saran sebagaimana yang dipaparkan pada bab 6. Rancangan penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Rancangan Penelitian (Pressman, 2001)

3.4.1. Analisis Kebutuhan Sistem

Analisis kebutuhan merupakan tahap untuk mengumpulkan data dan informasi yang dibutuhkan untuk membangun sistem. Data tersebut dikelompokkan menjadi kebutuhan fungsional dan kebutuhan non fungsional. Untuk memahami sifat program yang akan dibangun, maka harus memahami informasi yang dibutuhkan untuk perangkat lunak, fungsi yang diperlukan, alur, kinerja dan *interface* dari program yang akan dibangun (Pressman, 2001).

1. Teknik Pengumpulan Data

   Beberapa teknik untuk pengumpulan data pada penelitian ini antara lain:

a. Wawancara yang merupakan teknik di mana dilakukan tanya-jawab kepada para narasumber yang telah tercantum di subbab 3.2, yaitu tempat dan waktu penelitian.
b. Studi literatur yang merupakan teknik melakukan peninjauan secara cermat untuk proses pengumpulan data dan studi literatur dari berbagai sumber seperti buku, jurnal dan internet.

2. Teknik Pengolahan Data

Data yang dibutuhkan untuk membangun sistem pada penelitian ini antara lain data penggunaan trafo sebagai data utama untuk prediksi menggunakan metode ANN dan ACO, data penggunaan trafo sebagai data masukan untuk perhitungan metode ANN dan ACO dan data kepegawaian PT. PLN Area Jember sebagai data untuk membedakan hak akses yang dapat menggunakan sistem pengklasifikasi ini.

3.4.2. Perancangan Sistem

Pada tahap ini, dilakukan perancangan dengan menggunakan UML yang mendukung konsep pemodelan pemrograman berbasis OOD seperti yang akan diterapkan pada tahap implementasi sistem. Hasil perancangan sistem yang didapat antara lain *business process*, *use case* diagram, skenario, *sequence diagram*, *collaboration diagram*, *state diagram*, *class diagram*, dan *entity relationship diagram* atau ERD.

3.4.3. Implementasi Sistem

Pada tahap ini, dilakukan implementasi dari perancangan sistem sehingga menghasilkan kode-kode program untuk sistem ini. Sistem dibangun menggunakan bahasa Java. Sistem pengelolaan basis data yang digunakan adalah MySQL.

3.4.4. Pengujian Sistem

Pada tahap ini, dilakukan pengujian pada sistem secara keseluruhan. Metode yang digunakan untuk pengujian sistem adalah *black box* dan *white box*. *White box* merupakan metode untuk membaca alur logika yang berjalan pada sistem.

Sedangkan *black box* merupakan metode yang dilakukan pada fungsionalitas sistem tanpa pengetahuan *source code* sistem tersebut.

3.4.5. Pemeliharaan Sistem

Pada tahap ini, dilakukan pencarian kesalahan-kesalahan atau *bug* pada sistem yang telah dibuat di mana kesalahan-kesalahan tersebut muncul dikarenakan kondisi-kondisi tertentu seperti perubahan kebutuhan-kebutuhan dan performa sistem. Jika muncul kesalahan-kesalahan tersebut, maka sistem akan diubah pada tahap tertentu yang terus berkelanjutan hingga tahap ini sehingga kesalahan tersebut dapat teratasi.

BAB 4. PENGEMBANGAN SISTEM

Pengembangan sistem ini dilakukan dengan menggunakan model *waterfall*. Model ini merupakan metodologi pengembangan perangkat lunak yang mengusulkan pendekatan kepada perangkat lunak sistematik dan sekuensial yang mulai pada tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan.

4.1. Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak dalam penelitian ini yaitu dengan cara mengidentifikasi permasalahan yang ada untuk kemudian dicatat dan dijadikan bahan untuk mulai membangun sistem informasi peramalan jumlah konsumsi dan harga cabai. Analisis kebutuhan yang dilakukan meliputi proses pengumpulan data kebutuhan fungsional dan kebutuhan non-fungsional. Adapun data yang terkumpul dari proses analisis kebutuhan adalah data persediaan dan data penggunaan trafo tiap bulan. Data yang diperoleh dari hasil analisis kebutuhan adalah sebagai berikut:

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional untuk mengembangkan sistem informasi trafo adalah sebagai berikut:

1. Sistem menggunakan fitur *login* sebagai menu verifikasi pada pengguna sesuai dengan hak akses yang dimiliki.
2. Sistem dapat mengelola (tambah, ubah, hapus) dataset persediaan trafo.
3. Sistem dapat menampilkan tabel persediaan trafo.
4. Sistem dapat mengelola (tambah, ubah, tampil) data kategori trafo.
5. Sistem dapat melakukan penghitungan metode ANN dengan ACO sebagai metode pembelajaran sesuai menggunakan dataset persediaan trafo sebagai sumber.
6. Sistem dapat menampilkan hasil prediksi persediaan trafo hasil perhitungan dari metode ANN dengan ACO sebagai metode pembelajaran.
7. Sistem mampu mengelola data user (create, view, edit, dan delete).

4.1.2 Kebutuhan Non-fungsional

Kebutuhan non-fungsional yang menjadi pendukung kebutuhan fungsional pada pengembangan sistem informasi trafo adalah sebagai berikut:

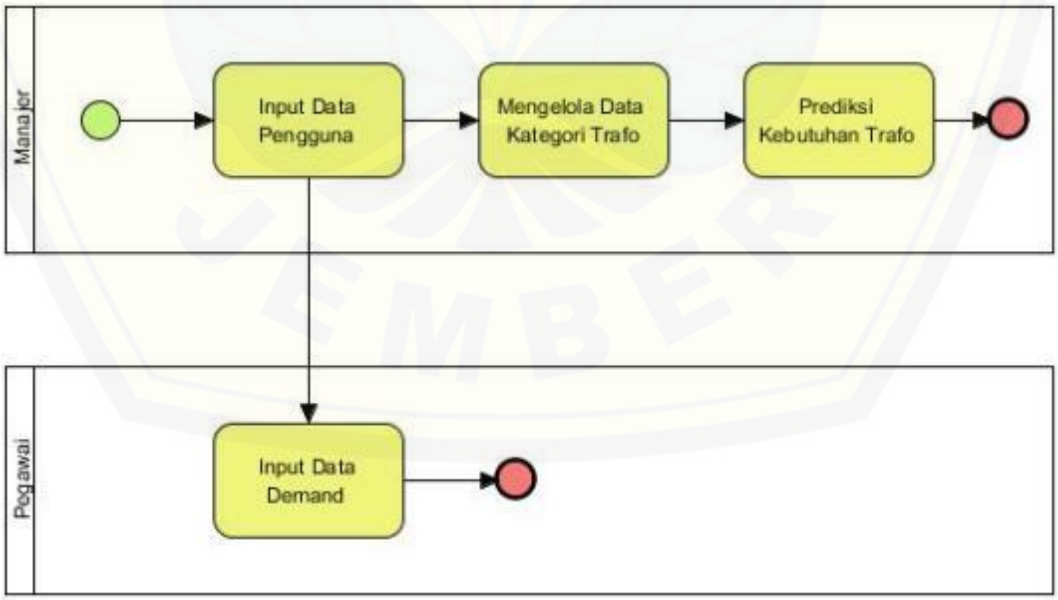
1. Tampilan dan bahasa komunikasi sistem mudah dimengerti oleh pengguna untuk memberikan kenyamanan pemakaian dan memudahkan pengoperasian.
2. Sistem menggunakan username dan password untuk autentifikasi akses admin terhadap sistem.

4.2. Desain Sistem

Tahapan berikutnya adalah desain sistem menggunakan *Unified Modeling Language* (UML) yang dirancang menggunakan konsep *Object-Oriented Programming* (OOP). Berikut pemodelan UML yang digunakan antara lain:

4.2.1 *Business Process*

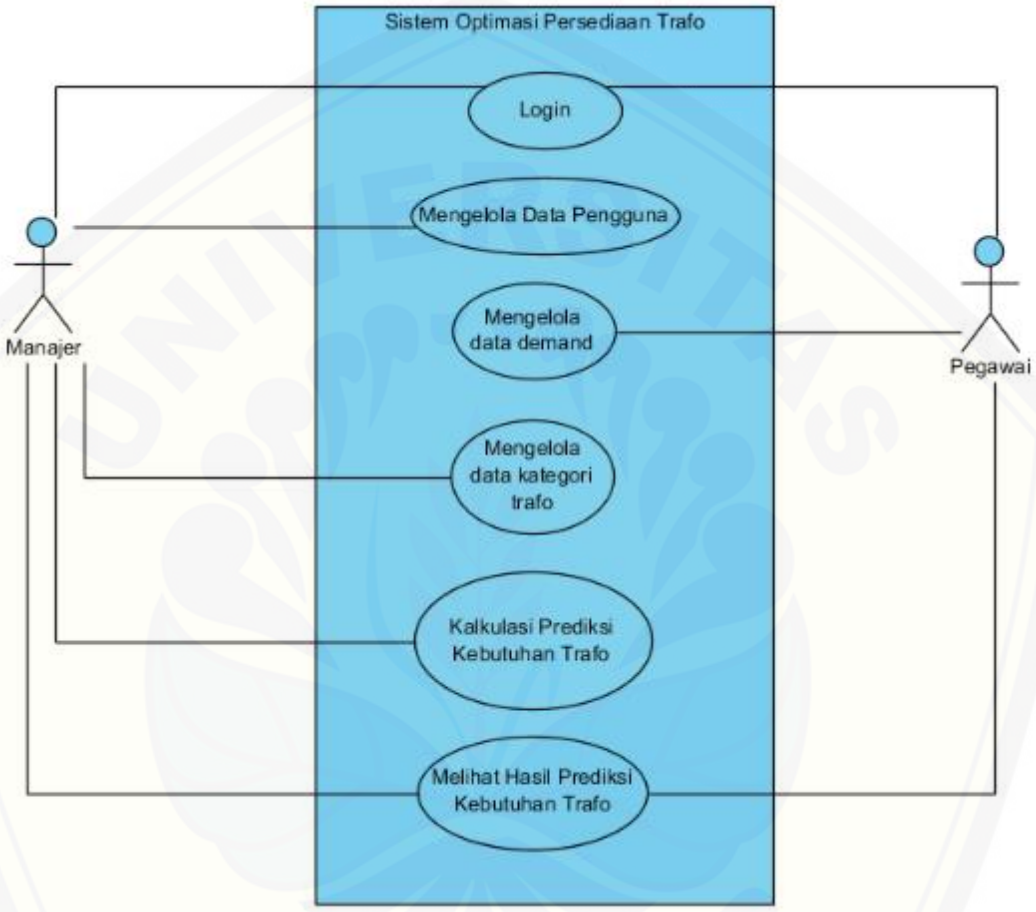
Business process merupakan diagram yang menggambarkan proses dari sebuah sistem yang meliputi data apa yang diperlukan lalu data diolah untuk menghasilkan *output* yang diinginkan. *Business process* dalam pengembangan sistem optimasi persediaan trafo dapat dilihat pada gambar 4.1.



Gambar 4. 1 *Business Process* Sistem Informasi Trafo

4.2.2 *Use case Diagram*

Menggambarkan fitur-fitur yang tersedia pada aplikasi yang dibangun dan hak akses setiap aktor yang terlibat. *Use case diagram* ditunjukkan pada gambar 4.2 yang terdapat 2 aktor yang dapat mengakses sistem yaitu Manajer dan Pegawai.



Gambar 4. 2 *Use case Diagram*

Fitur-fitur yang terdapat pada sistem beserta aktor yang berhak mengakses fitur tersebut berdasarkan *use case diagram* dijelaskan pada tabel berikut.

Tabel 4. 1 Definisi Aktor

No.	**Aktor**	**Deskripsi**
1	Manajer	Aktor manajer dapat mengelola data *demand*, kategori trafo, melakukan kalkulasi kebutuhan trafo, serta melihat hasil prediksi .

2	Pegawai	Aktor ini dapat mengelola data *demand*, kategori trafo, serta melihat hasil prediksi kebutuhan trafo.

Tabel 4. 2 Definisi *Use Case*

No.	***Use case***	**Penjelasan**
UC-01	*Login*	Merupakan *use case* yang berfungsi untuk masuk dalam sistem optimasi persediaan trafo
UC-02	Mengelola Data Pengguna	Merupakan *use case* yang menggambarkan proses menampilkan, mengubah, menambah, dan menghapus data pengguna yang berhak mengakses aplikasi ini. Setiap pengguna memiliki hak akses berbeda
UC-03	Mengelola data *demand* trafo	Merupakan *use case* yang menggambarkan proses menampilkan, mengubah, menambah, dan menghapus data penggunaan trafo tiap bulan
UC-04	Mengelola data kategori trafo	Merupakan *use case* yang menggambarkan proses menampilkan, mengubah, menambah, dan menghapus data kategori trafo
UC-05	Kalkulasi Prediksi Kebutuhan Trafo	Merupakan *use case* yang berfungsi untuk melakukan kalkulasi untuk prediksi kebutuhan trafo 1 tahun kedepan
UC-06	Melihat Hasil Prediksi Kebutuhan Trafo	Merupakan *use case* yang berfungsi untuk melihat hasil prediksi kebutuhan trafo 1 tahun kedepan.

4.2.3 *Scenario Diagram*

Scenario digunakan untuk menjelaskan cara kerja sistem berdasarkan tugas *user* yang terdapat pada *use case* diagram. *Scenario* terdiri dari nama *use case*, aksi aktor dan reaksi sistem. Berikut ini *scenario* dari optimasi distribusi cabai:

1. *Scenario* Mengelola Data *Demand* Trafo

Penjelasan urutan reaksi aktor dan reaksi sistem pada *scenario* normal dan *scenario* alternatif *use case* mengelola data *Demand* Trafo di Area Jember dapat dilihat di tabel 4.3.

Tabel 4. 3 *Scenario* Mengelola Data *Demand* Trafo

Nomor *Use case*	**UC-03**
Nama	Mengelola data *demand* trafo
Aktor	Manajer
Precondition	Manajer memilih menu *View*
Postcondition	Manajer berhasil melihat halaman *demand* trafo di Area Jember
***SCENARIO* NORMAL** **"Melihat Data *Demand* Trafo"**	
Aktor	**Sistem**
1. Memilih menu Permintaan	
	2. Menampilkan pilihan *Combo Box* kategori
3. Memilih kategori 4. Mengklik tampilkan	
	5. Menampilkan data penggunaan trafo sesuai kategori yang dipilih
***SCENARIO* ALTERNATIF** **"Menambahkan Data *Demand* Trafo"**	
Aktor	**Sistem**
6. Memilih bulan 7. Mengisi tahun 8. Mengisi Jumlah 9. Mengklik tombol "Tambah"	

	10. Menyimpan inputan ke database 11. Menampilkan inputan ke tampilan tabel

2. *Scenario* Mengelola Data Kategori Trafo

Penjelasan *scenario* mengelola data trafo yang merupakan urutan reaksi aktor dan reaksi sistem pada *scenario* normal dan *scenario* alternatif mengelola data kategori Trafo di Area Jember dapat dilihat di Lampiran A.1 pada tabel A.1.

3. *Scenario* Mengelola Data Pengguna

Penjelasan *scenario* mengelola data pengguna yang merupakan urutan reaksi aktor dan reaksi sistem pada *scenario* normal dan *scenario* alternatif *use case* mengelola data pengguna sistem optimasi persediaan trafo dapat dilihat di Lampiran A.2 pada tabel A.2.

4. *Scenario* Kalkulasi Prediksi Kebutuhan Trafo

Penjelasan *scenario* untuk melakukan kalkulasi kebutuhan trafo merupakan urutan reaksi aktor dan reaksi sistem pada *scenario* normal dan *scenario* alternatif *use case* kalkulasi prediksi kebutuhan Trafo di Area Jember dapat dilihat di Lampiran A.3 pada tabel A.3.

5. *Scenario* Melihat Hasil Prediksi Kebutuhan Trafo

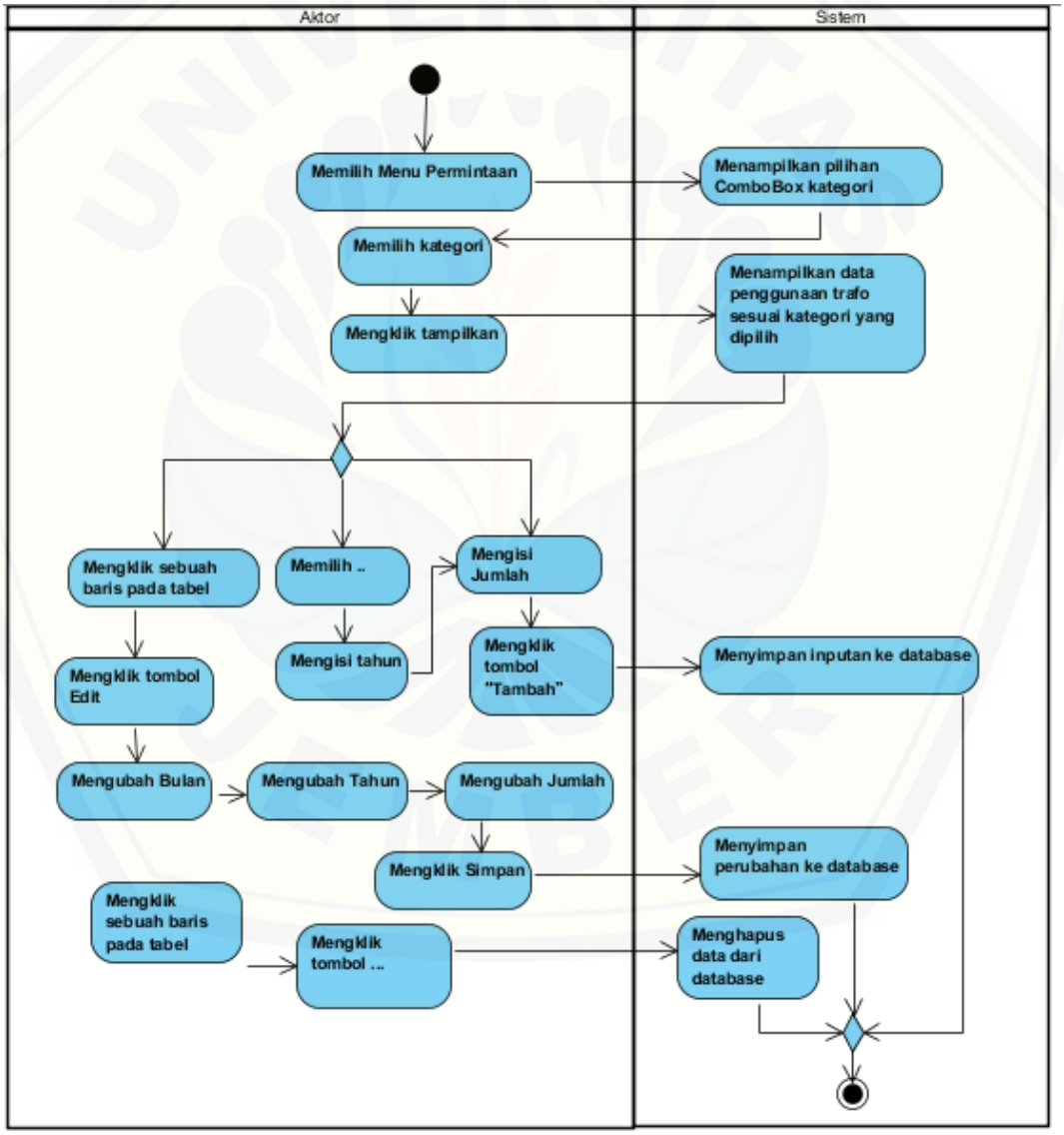
Penjelasan *scenario* untuk melihat hasil prediksi kebutuhan persediaan trafo merupakan urutan reaksi aktor dan reaksi sistem pada *scenario* normal *use case* melihat hasil prediksi kebutuhan trafo di Area Jember dapat dilihat di Lampiran A.4 pada tabel A.4.

4.4.5. *Activity Diagram*

Activity Diagram digunakan untuk menggambarkan urutan aktivitas dalam sebuah proses. Aktivitas tersebut sesuai dengan *scenario* yang berisi tugas *user* dan reaksi sistem dan digambarkan dalam bentuk diagram.

1. *Activity Diagram* Mengelola Data *Demand* Trafo

Activity diagram untuk mengelola data *demand* trafo menggambarkan urutan aktivitas dalam sebuah proses yang dilakukan oleh aktor dan reaksi sistem. Aktivitas untuk untuk mengelola data *demand* trafo dapat dilihat pada gambar 4.4



Gambar 4. 3 *Activity Diagram* Mengelola Data *Demand* Trafo

2. *Activity Diagram* Mengelola Data Kategori Trafo

Activity ini diakses oleh manajer dan pegawai digunakan untuk mengelola data konsumsi kategori trafo. *Activity Diagram* dapat dilihat di Lampiran B.1 pada gambar B.1.

3. *Activity Diagram* Mengelola Data Pengguna

Activity ini diakses oleh manajer dan pegawai digunakan untuk mengelola data pengguna sistem informasi trafo. *Activity Diagram* dapat dilihat di Lampiran B.2 pada gambar B.2.

4. *Activity Diagram* Kalkulasi Prediksi Kebutuhan Trafo

Activity ini diakses oleh manajer digunakan untuk menjalankan kalkulasi prediksi kebutuhan trafo. *Activity Diagram* dapat dilihat di Lampiran B.3 pada gambar B.3.

5. *Activity Diagram* Melihat Hasil Prediksi Kebutuhan Trafo

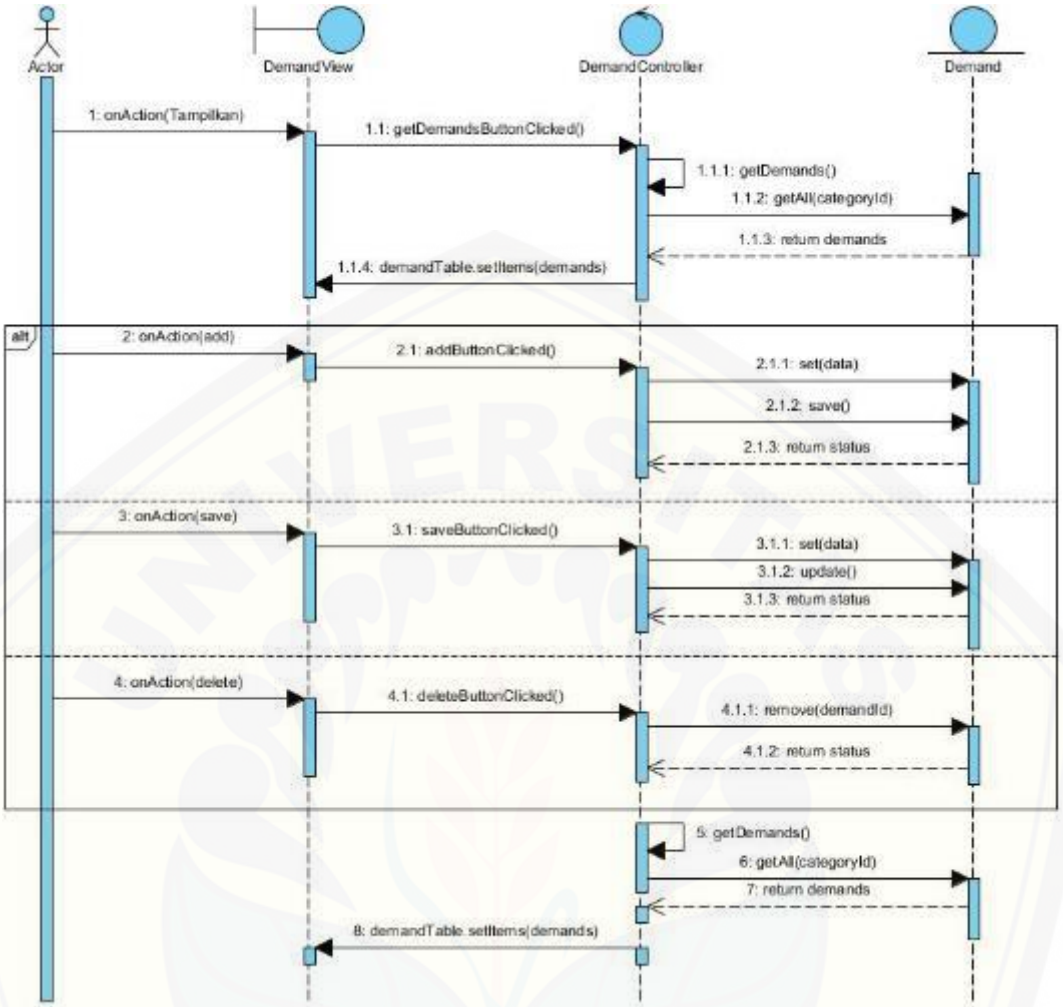
Activity ini diakses oleh manajer digunakan untuk melihat hasil kalkulasi prediksi kebutuhan trafo. *Activity Diagram* dapat dilihat di Lampiran B.4 pada gambar B.4.

4.2.4 *Sequence Diagram*

Sequence Diagram digunakan untuk menunjukkan interaksi antar objek pada sebuah sistem berupa pesan yang digambarkan terhadap waktu. *Sequence* merupakan *blue print* bagi *programmer*.

1. *Sequence Diagram* Mengelola Data *Demand* Trafo

Sequence ini diakses oleh manajer dan pegawai digunakan mengelola data konsumsi kategori trafo menjelaskan urutan untuk menambah, mengubah, dan menghapus data permintaan trafo dalam sistem. *Sequence Diagram* dapat dilihat pada gambar berikut:



Gambar 4. 4 *Sequence Diagram* Mengelola Data *Demand* Trafo

2. *Sequence Diagram* Mengelola Data Kategori Trafo

Sequence ini diakses oleh manajer dan pegawai digunakan mengelola data konsumsi kategori trafo menjelaskan urutan untuk menambah, mengubah, dan menghapus data konsumsi kategori trafo dalam sistem. *Sequence Diagram* dapat dilihat di Lampiran C.1 pada gambar C.1.

3. *Sequence Diagram* Mengelola Data Pengguna

Sequence ini diakses oleh manajer digunakan mengelola data pengguna menjelaskan urutan untuk menambah, mengubah, dan menghapus data pengguna dalam sistem. *Sequence Diagram* dapat dilihat di Lampiran C.2 pada gambar C.2.

4. *Sequence Diagram* Kalkulasi Prediksi Kebutuhan Trafo

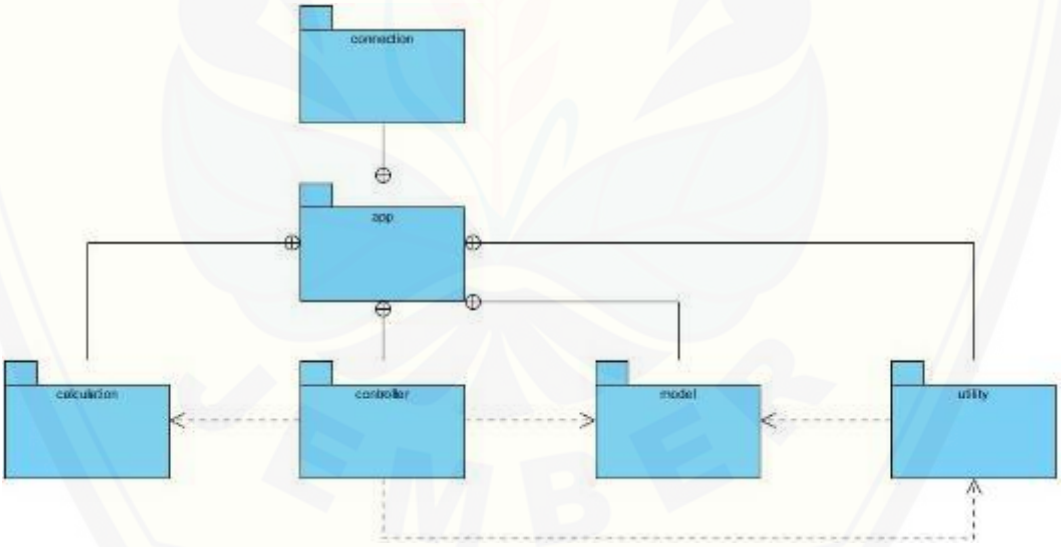
Sequence ini diakses oleh manajer digunakan menjalankan kalkulasi prediksi kebutuhan trafo menjelaskan urutan untuk melakukan kalkulasi prediksi kebutuhan trafo dalam sistem. *Sequence Diagram* dapat dilihat di Lampiran C.3 pada gambar C.3.

5. *Sequence Diagram* Melihat Hasil Prediksi Kebutuhan Trafo

Sequence ini diakses oleh manajer digunakan melihat hasil kalkulasi prediksi kebutuhan trafo menjelaskan urutan untuk melihat hasil prediksi kebutuhan trafo dalam sistem. *Sequence Diagram* dapat dilihat di Lampiran C.4 pada gambar C.4.

4.4.6. *Package Diagram*

Package Diagram menggambarkan hubungan *package* satu sama lain seperti pewarisan, asosiasi, dependensi dan lain-lain pada suatu sistem.



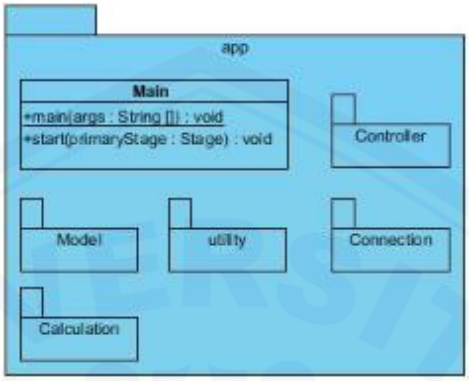
Gambar 4. 5 *Package Diagram*

4.4.7. *Class Diagram*

Class Diagram menggambarkan struktur dan deskripsi *class* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dependensi dan lain-

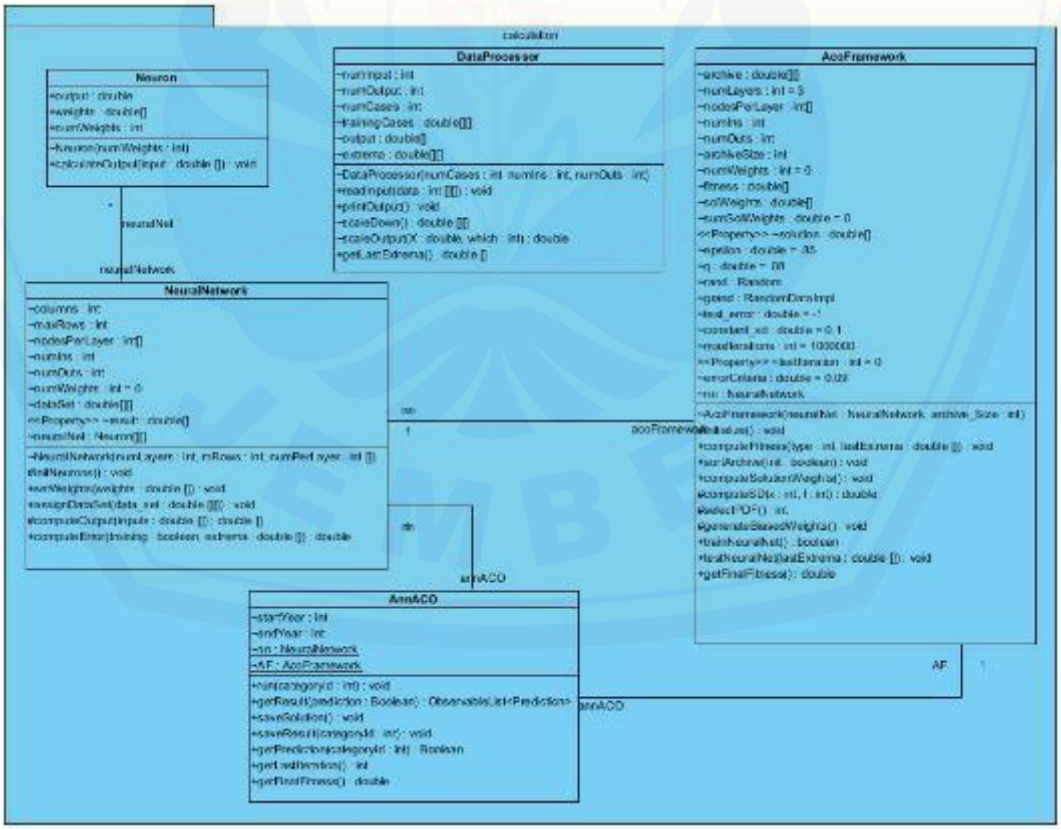
lain. *Class diagram* pada pengembangan sistem ini dibagi kedalam beberapa *package*., yang diantaranya terdiri dari:

1. *App Package*



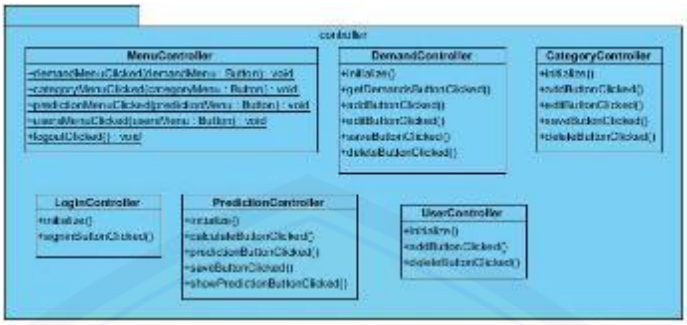
Gambar 4. 6 Class Diagram App Package

2. *Calculation Package*



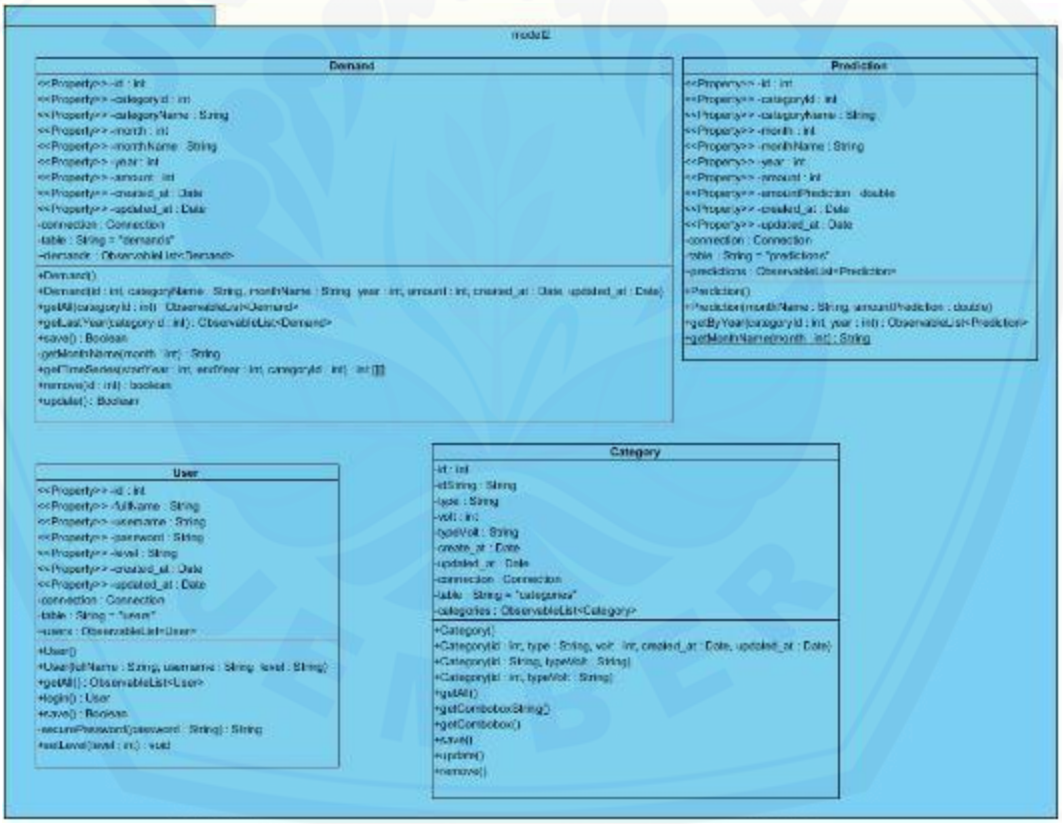
Gambar 4. 7 Class Diagram Calculation Package

3. *Controller Package*



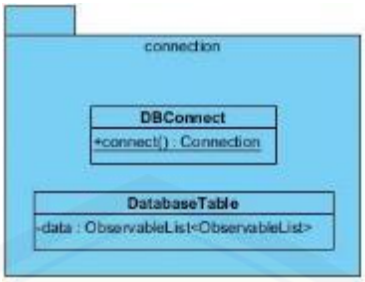
Gambar 4. 8 Class Diagram Controller Package

4. *Model Package*



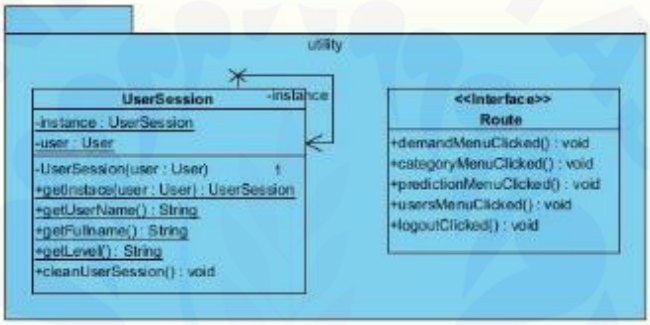
Gambar 4. 9 Class Diagram Model Package

5. *Connection Package*



Gambar 4. 10 Class Diagram Connection Package

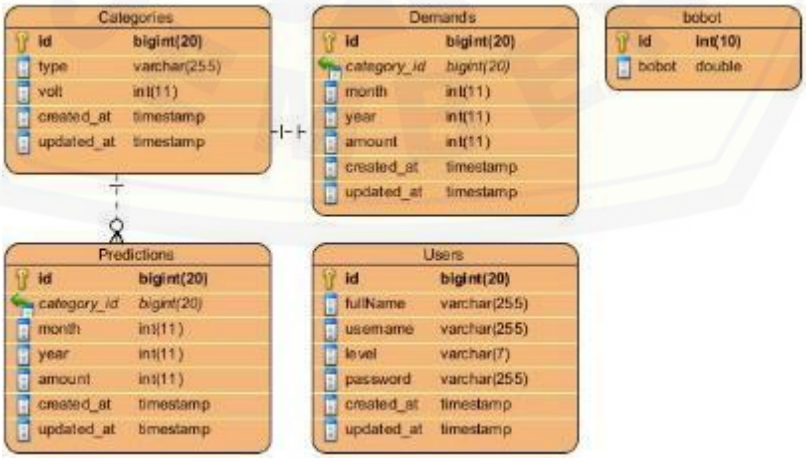
6. *Utility Package*



Gambar 4. 11 Class Digram Utility Package

4.4.8. *Entity Relationship Diagram*

Entity Relationship Diagram menggambarkan struktur *database* yang akan dibangun pada sistem.



Gambar 4. 12 Entity Relationship Diagram

4.3. Penulisan Kode Program

Tahap ini merupakan proses pembuatan aplikasi, yaitu merupakan proses implementasi dari hasil analisa kebutuhan yang dilakukan pada tahap perencanaan, analisa dan desain sistem. Penulisan kode program dilakukan menggunakan bahasa pemrograman *Java* dan menggunakan *MySQL Database*.

4.3.1. Kode Login

Penulisan kode login dalam sistem optimasi persediaan trafo terletak di class *LoginView* pada *view* dan *LoginController* pada *controller*. Penulisan kode demand dapat dilihat pada lampiran D.1 (Kode Program)

4.3.2. Kode Pengguna

Penulisan kode pengguna dalam sistem optimasi persediaan trafo terletak di *class UsersView* pada *view*, *class User* pada *model*, dan *UserController* pada *controller*. Penulisan kode demand dapat dilihat pada lampiran D.2 (Kode Program)

4.3.3. Kode Demand

Penulisan kode demand dalam sistem optimasi persediaan trafo terletak di *class DemandView* pada *view*, *class Demand* pada *model*, dan *DemandController* pada *controller*. Penulisan kode demand dapat dilihat pada lampiran D.3 (Kode Program)

4.3.4. Kode Kategori

Penulisan kode kategori dalam sistem optimasi persediaan trafo terletak di *class CategoryView* pada *view*, *Category* pada *model*, dan *CategoryController* pada *controller*. Penulisan kode kategori dapat dilihat pada lampiran D.4 (Kode Program)

4.3.5. Kode Prediksi

Penulisan kode prediksi dalam sistem optimasi persediaan trafo terletak di *class AnnACO, ACOFramework, DataProcessor, NeuralNetwor*k dan *Neuron* pada

controller. Penulisan kode prediksi dapat dilihat pada lampiran D.5 (Kode Program)

4.4. Pengujian Sistem

Pengujian dan evaluasi dimaksudkan untuk mengetahui sejauh mana sistem yang dibuat ini dapat berfungsi sesuai dengan proses transaksi dalam optimasi distribusi ini nanti sesuai dengan yang diharapkan oleh pada pengguna. Pada tahap ini dilakukan uji coba terhadap sistem optimasi untuk menemukan kesalahan-kesalahan yang mungkin terjadi serta melakukan perbaikan untuk lebih menyempurnakan kinerja desktopsite tersebut. Pengujian dilakukan dengan 2 metode, yaitu *White Box* dan *Black Box*. Pengujian ii menggunakan pengujian jalur dasar (basic *path testing*) yang didalamnya terdapat beberapa tahapan pengujian antara lain pembuatan diagram alir, penentuan jalur independen, penghitungan kompleksitas siklomatik, dan *test cas*e.

4.4.1. *Black Box Testing*

Pengujian Black Box melibatkan pengguna/*user*, dimana hanya memperhatikan fungsionalitas yang berkaitan dengan masukan/keluaran (I/O) apakah sesuai dengan sistem yang dijalankan

Tabel 4. 4 *Black Box Testing*

No	Menu	Fungsi	Aksi	Hasil	Ket
1	Login	Digunakan untuk masuk ke sistem	Mengisi Username dan Password pada form login kemudian menekan tombol login	Login berhasil dan menampilkan halaman home bagian pergudangan	√

2	Permintaan	Menampilkan data permintaan tiap tahun	Memilih kategori dan mengklik tombol tampilkan	Menampilkan data permintaan tiap tahun	√
		Menambah data permintaan	Memilih kategori lalu mengisi bulan, tahun, jumlah trafo. Mengklik tombol tambah	Menyimpan data ke database lalu menampilkan hasilnya	√
		Mengubah data permintaan	Memilih data pada tabel, mengklik tombol edit, mengubah bulan, tahun, jumlah trafo, mengklik tombol simpan	Menyimpan perubahan data ke database lalu menampilkan hasilnya	√
		Menghapus data permintaan	Memilih data pada tabel, mengklik tombol hapus	Menghapus data dari database	√
3	Kategori	Menampilkan data kategori	Mengklik menu kategori	Menampilkan data kategori	√
		Menambah data kategori	Memilih kategori lalu mengisi tipe dan volt. Mengklik tombol tambah	Menyimpan data ke database lalu menampilkan hasilnya	√

		Mengubah data kategori	Memilih data pada tabel, mengklik tombol edit, mengubah tipe dan volt, mengklik tombol simpan	Menyimpan perubahan data ke database lalu menampilkan hasilnya	√
		Menghapus data kategori	Memilih data pada tabel, mengklik tombol hapus	Menghapus data dari database	√
4	Pengguna	Menampilkan data pengguna	Mengklik menu pengguna	Menampilkan data pengguna	√
		Menambah data pengguna	Memilih kategori lalu mengisi nama lengkap, username, level, password. Mengklik tombol tambah	Menyimpan data ke database lalu menampilkan hasilnya	√
		Mengubah data pengguna	Memilih data pada tabel, mengklik tombol edit, mengubah nama lengkap, username, level, password, mengklik tombol simpan	Menyimpan perubahan data ke database lalu menampilkan hasilnya	√

		Menghapus data pengguna	Memilih data pada tabel, mengklik tombol hapus	Menghapus data dari database	√
5	Prediksi	Melakukan Kalkulasi	Memilih kategori, mengklik kalkulasi	Melakukan kalkulasi dan menampilkan hasilnya	√
		Menampilkan Prediksi tahun depan	Mengklik tombol prediksi	Menampilkan hasil prediksi	√
		Menyimpan hasil prediksi	Mengklik tombol simpan	Menyimpan hasil prediksi ke database	√
		Menampilkan data prediksi	Mengklik tombol tampilkan	Menampilkan data prediksi	√

BAB 6. PENUTUP

Bab ini menjelaskan kesimpulan dan saran yang didapatkan dalam penelitian dan pengembangan sistem informasi persediaan trafo mengunakan metode *ant colony optimization.*

6.1. Kesimpulan

Kesimpulan dari pengembangan sistem ini adalah sebagai berikut:

1. Metode jaringan syaraf tiruan dan *ant colony optimization* sebagai metode pembelajaran dapat digunakan untuk menghitung prediksi jumlah penggunaan trafo pada bulan berikutnya.
2. Hasil prediksi jumlah penggunaan trafo pada bulan berikutnya dalam satu tahun dapat dijadikan patokan dalam menentukan kebutuhan. Pada tahun 2018 dengan data kebutuhan trafo pada bulan januari berjumlah 13 buah. Prediksi perhitungan pada tahun 2018 menggunakan metode *ant colony optimization* menghasilkan 239 unit. Adapun hasil peramalan memiliki tingkat error sebesar 30 unit. Pada *mean absolute deviation* sebesar 3.5, sedangkan pada *square of error* bernilai 16.667. Secara umum hasil penilaian persentase kesalahan absolut rata-rata sebesar 23.28%. Maka, data peramalan yang dilakukan memiliki tingkat akurasi yang cukup tinggi.
3. Metode jaringan syaraf tiruan dan *ant colony optimization* sebagai metode pembelajaran dapat diterapkan dalam sistem berbasis Java guna mempermudah proses pengembangan sistem.

6.2. Saran

Beberapa saran berikut diharapkan dapat memberikan perbaikan dalam penelitian selanjutnya, yaitu:

1. Penggunaan objek penelitian untuk menghitung menggunakan metode jaringan syaraf tiruan sebaiknya ditambah dengan data pasti terkait cuaca dan musim tiap bulan, sehingga perhitungan memiliki nilai akurasi yang tinggi.

2. Untuk memperoleh hasil prediksi yang lebih baik data yang dikumpulkan memiliki lebih banyak variabel untuk membantu proses perhitungan menggunakan metode jaringan syaraf tiruan.

DAFTAR PUSTAKA

Bakirtzis, A., & Petidis, V. (1996). "A Neural Network Short-Term Load Forecasting Model for Greek Power Systems" IEEE Transactions on Power Systems 11. *Journal*, 858-863.

Buana, M. I. (2014). Ant Colony Optimization Dalam Penyelesaian Travelling Salesman Problem Menggunakan Matlab. *Jurnal*, 1-14.

de Myttenaere, B. G. (2016). Mean absolute percentage error for regression models. Dalam *Neurocomputing* (hal. 3). Paris: Universit´e Paris 1 Panth´eon.

Departemen Pendidikan Indonesia. (2008). *Kamus Besar Bahasa Indonesia.* Jakarta: Balai Pustaka.

Gaspersz, & Vincent. (1998). *Manajemen Produksi Total, Strategi Peningkatan Produktivitas Bisnis Global.* Jakarta: Gramedia Pustaka Utama,.

Greene, M. J., & Gordon, D. M. (2007). *Structural Complexity of Chemical Recognition Cues Affects The Perception of Group Membership in The Ants Linephithema Humile and Aphaenogaster Cockerelli.* Journal of Experimental Biology.

Hermawan, A. (2006). *Jaringan Syaraf Tiruan Teori dan Aplikasi.* Jogjakarta: Andi Publisher.

Hu, X., Zhang, J., & Li, Y. (2008). *Orthogonal methods based ant colony search for solving continuous optimization problems.* Springer.

Husna, J., Pelawi, Z., & Yusniati. (2008). Menentukan Indeks Saidi dan Saifi Pada Saluran Udara Tegangan Menengah Di PT. PLN Wilayah NAD Cabang Langsa. *Jurnal*, 1-17.

Kader, G. (1999). Means and MADS. Dalam *Mathematics Teaching in the Middle School* (hal. 398-403). Virginia: National Council of Teachers of Mathematics.

Kho, D. K., Suyanto, & Baizal, Z. A. (2011). *Prediksi Nilai Tukar IDR/USD Menggunakan Artificial Neural Networks Dengan Metode Pembelajaran Ant Colony Optimization.* Bandung: Universitas Telkom.

Kusumadewi, S., Hartati, S., & Harjoko, A. (2006). Fuzzy Multi-Attribute Decision Making (Fuzzy MADM). *Jurnal*, 78-79.

Nachrowi, & Hardius, U. (2006). *Pendekatan Popolerdan Praktis Ekonometrika Untuk Analisis Ekonomi dan Keuangan.* Jakarta: Lembaga Penerbit Fakultas Ekonomi Universitas Indonesia.

Pandian, A. (2013). *Training Neural Networks with Ant Colony Optimization.* Sacramento: California State University.

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach, Fifth Edition.* New York: McGraw-Hill Higher Education ©2001.

Suyanto. (2008). *Soft Computing.* Bandung: INFORMATIKA Bandung.

Wibowo, S. S. (2018). *Analisa Sistem Tenaga.* Malang: Polinema Press.

Wulandari, A. J., Mumpuni, S., & Irhamah. (2011). Optimasi Sistem Persediaan Multi-item di PT. AMIGO GROUP dengan Pendekatan Algoritma Genetika. *Jurnal*, 1-12.

LAMPIRAN

LAMPIRAN A. *Scenario*

A.1. *Scenario* Mengelola Data Kategori Trafo

Tabel A. 1 *Scenario* Mengelola Data Kategori Trafo

Nomor *Use case*	**UC-04**
Nama	Mengelola data kategori trafo
Aktor	Manajer
Precondition	Manajer memilih menu *View*
Postcondition	Manajer berhasil melihat halaman kategori trafo di Area Jember
***SCENARIO* NORMAL** **"Melihat Data Kategori Trafo"**	
Aktor	**Sistem**
1. Memilih menu Kategori	
	2. Menampilkan data kategori trafo
***SCENARIO* ALTERNATIF** **"Menambahkan Data Kategori Trafo"**	
Aktor	**Sistem**
4. Mengisi jenis 5. Mengisi volt 6. Mengklik tombol "Add"	
	7. Menyimpan inputan ke database 8. Menampilkan inputan ke tampilan tabel
***SCENARIO* ALTERNATIF** **"Menghapus Data Kategori Trafo"**	

Aktor	Sistem
4. Memilih baris pada tabel	
5. Mengklik tombol "Delete"	
	6. Menghapus data dari database 7. Menghapus data dari tampilan tabel

A.2. *Scenario* Mengelola Data Pengguna

Tabel A. 2 *Scenario* Mengelola Data Pengguna

Nomor *Use case*	**UC-04**
Nama	Mengelola data pengguna
Aktor	Manajer
Precondition	Manajer memilih menu *Users*
Postcondition	Manajer berhasil melihat halaman kategori trafo di Area Jember
***SCENARIO* NORMAL** **"Melihat Data Pengguna"**	
Aktor	**Sistem**
1. Memilih menu Pengguna	
	2. Menampilkan data pengguna
***SCENARIO* ALTERNATIF** **"Menambahkan Data Pengguna"**	
Aktor	**Sistem**
4. Mengisi nama lengkap 5. Mengisi username 6. Mengisi password	

7. Mengklik tombol "Add"	
	8. Menyimpan inputan ke database 9. Menampilkan inputan ke tampilan tabel
***SCENARIO* ALTERNATIF** **"Menghapus Data Kategori Trafo"**	
Aktor	**Sistem**
4. Memilih baris pada tabel	
5. Mengklik tombol "Delete"	
	6. Menghapus data dari database 7. Menghapus data dari tampilan tabel

A.3. *Scenario* Kalkulasi Prediksi Kebutuhan Trafo

Tabel A. 3 *Scenario* Kalkulasi Prediksi Kebutuhan Trafo

Nomor *Use case*	**UC-05**
Nama	Kalkulasi Prediksi Kebutuhan Trafo
Aktor	Manajer
Precondition	Manajer memilih menu *Prediction*
Postcondition	Manajer menjalankan kalkulasi prediksi kebutuhan trafo
***SCENARIO* NORMAL** **"Kalkulasi Prediksi Kebutuhan Trafo"**	
Aktor	**Sistem**
1. Memilih menu Prediksi	

	2. Menampilkan kategori trafo
3. Memilih kategori trafo 4. Mengklik tombol "Kalkulasi"	
	5. Menjalankan kalkulasi prediksi kebutuhan trafo 6. Menampilkan hasil prediksi

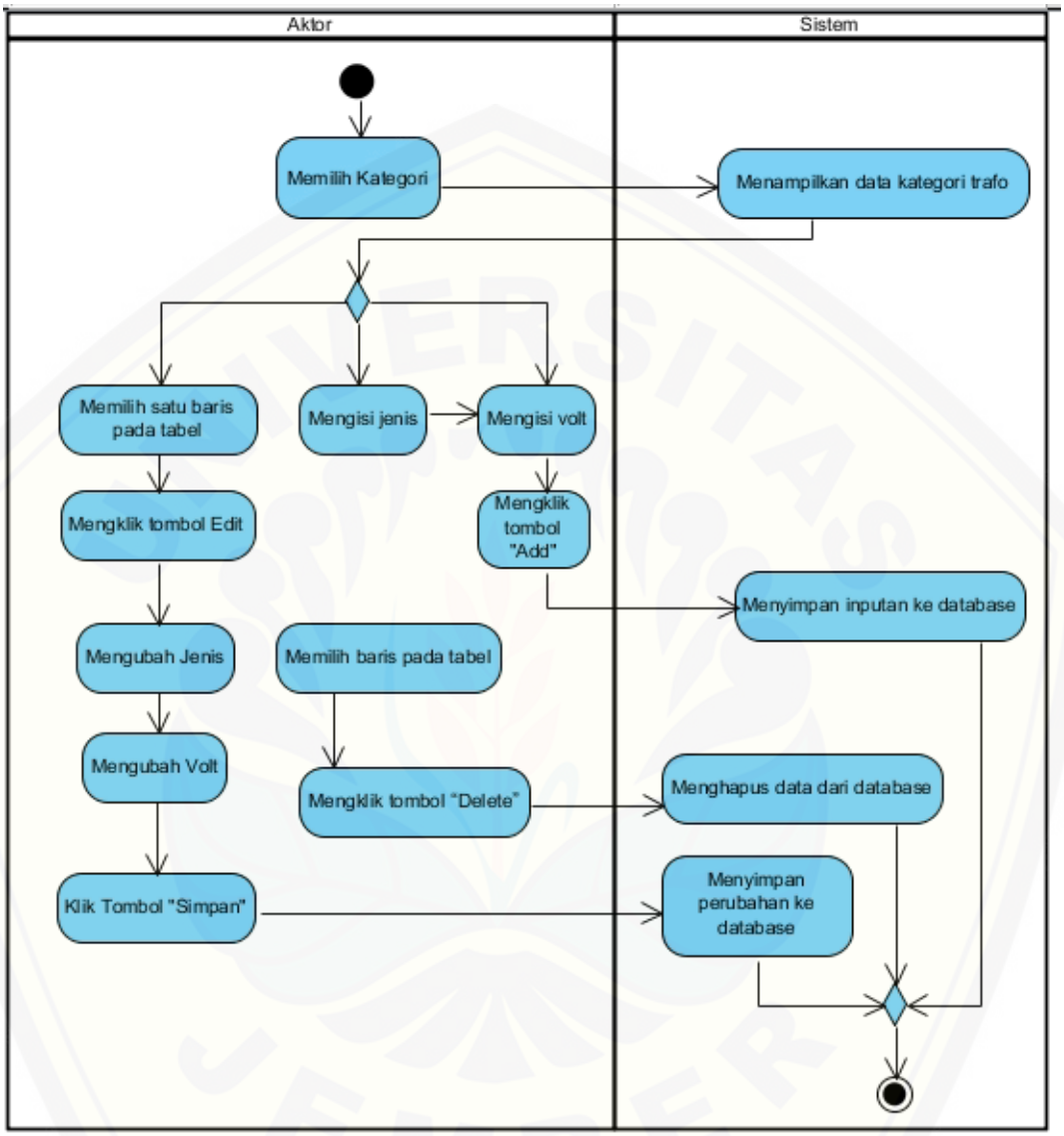
A.4. *Scenario* Melihat Hasil Prediksi Kebutuhan Trafo

Tabel A. 4 *Scenario* Melihat Hasil Prediksi Kebutuhan Trafo

Nomor *Use case*	**UC-05**
Nama	Kalkulasi Prediksi Kebutuhan Trafo
Aktor	Pegawai
Precondition	Pegawai memilih menu *Prediction*
Postcondition	Pegawai melihat hasil kalkulasi prediksi kebutuhan trafo
***SCENARIO* NORMAL** **"Melihat Hasil Prediksi Kebutuhan Trafo"**	
Aktor	**Sistem**
1. Memilih menu *Prediction* 2. Memilih Submenu *Result*	
	3. Menampilkan kategori trafo
4. Memilih kategori trafo 5. Mengklik tombol "Tampilkan"	
	6. Menampilkan hasil kalkulasi prediksi kebutuhan trafo

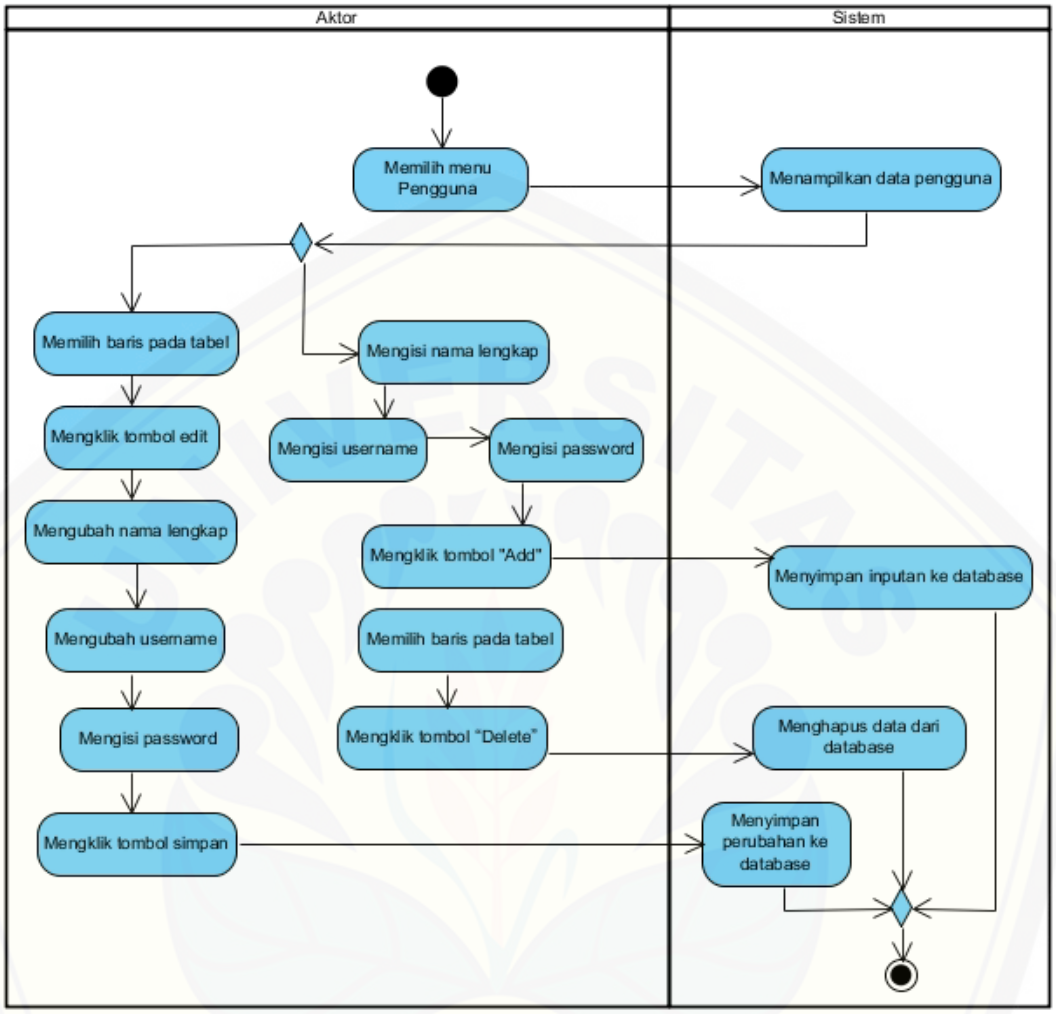
LAMPIRAN B. *Activity Diagram*

B.1 *Activity Diagram* Mengelola Data Kategori Trafo



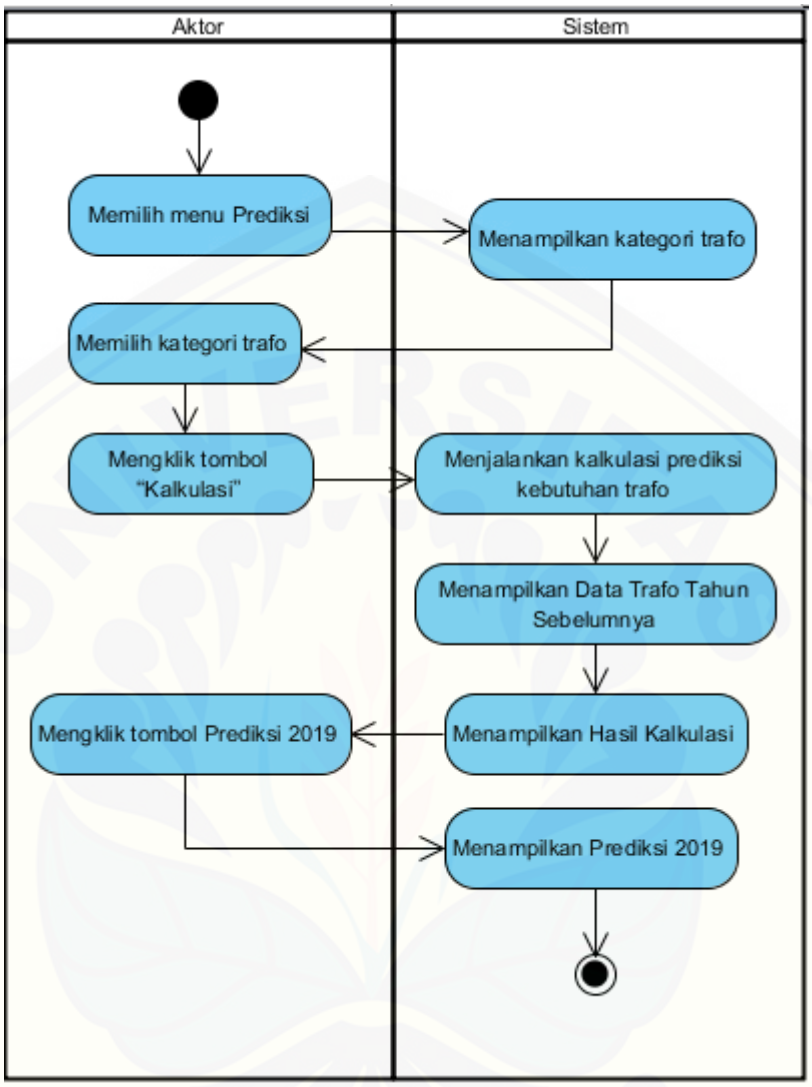
Gambar B. 1 *Activity Diagram* Mengelola Data Kategori Trafo

B.2 *Activity Diagram* Mengelola Data Pengguna



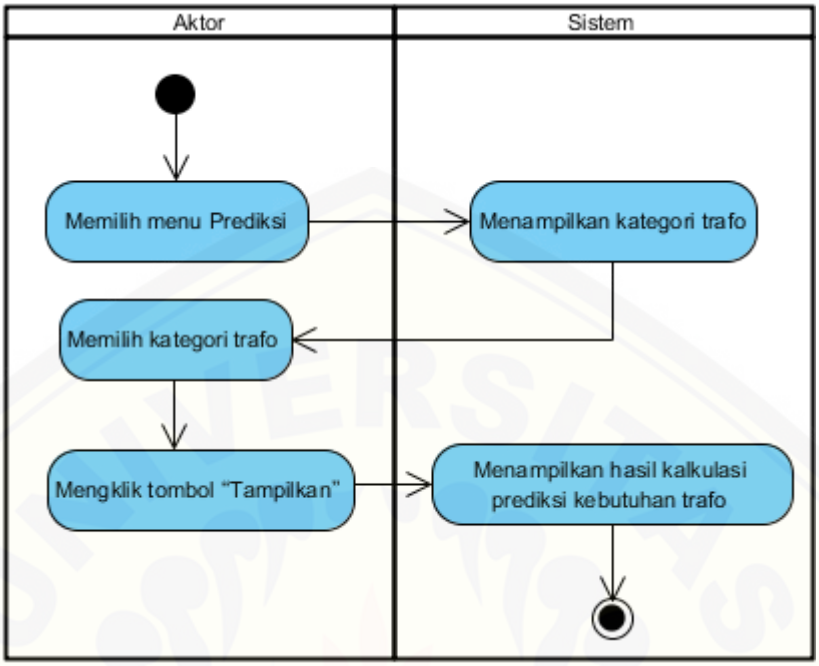
Gambar B. 2 *Activity Diagram* Mengelola Data Pengguna

B.3 *Activity Diagram* Kalkulasi Prediksi Kebutuhan Trafo



Gambar B. 3 *Activity Diagram* Kalkulasi Prediksi Kebutuhan Trafo

B.4 *Activity Diagram* Melihat Hasil Prediksi Kebutuhan Trafo

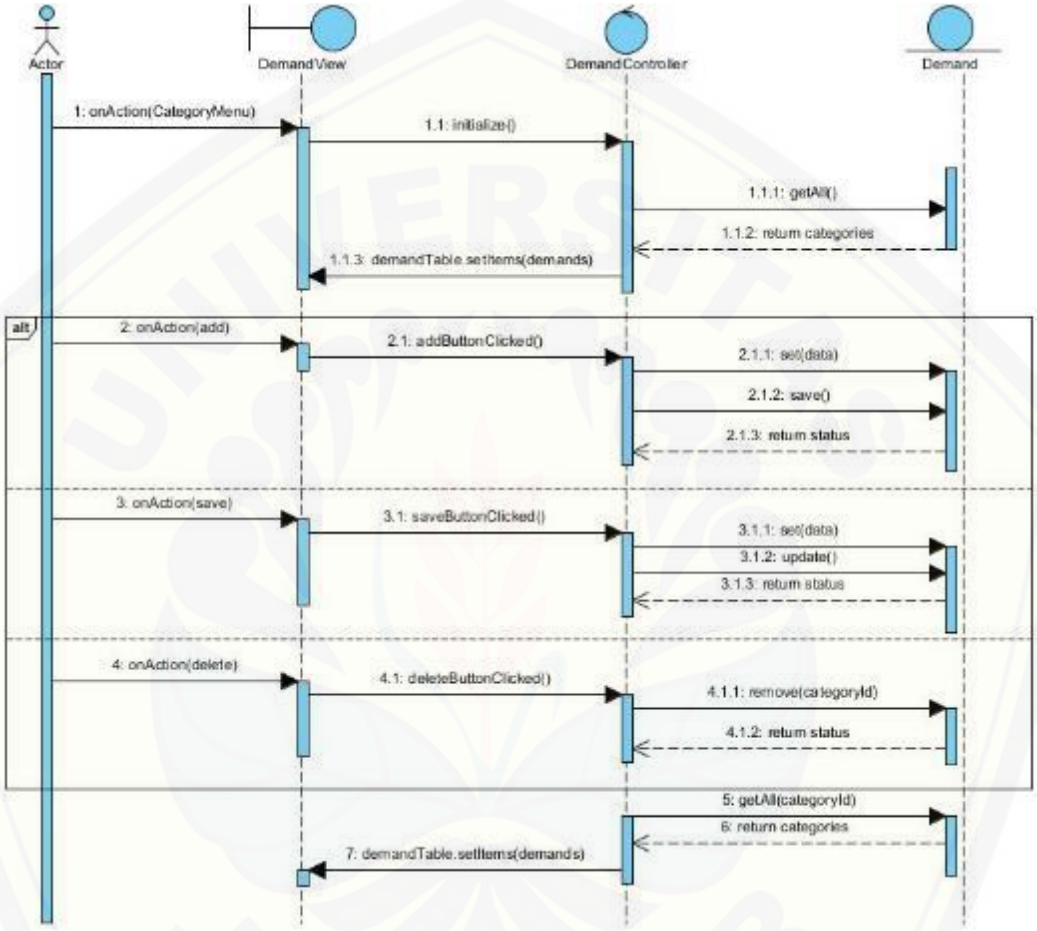


Gambar B. 4 *Activity Diagram* Melihat Hasil Prediksi Kebutuhan Trafo

LAMPIRAN C. *Sequence Diagram*

C.1. *Sequence Diagram* Mengelola Data Kategori Trafo

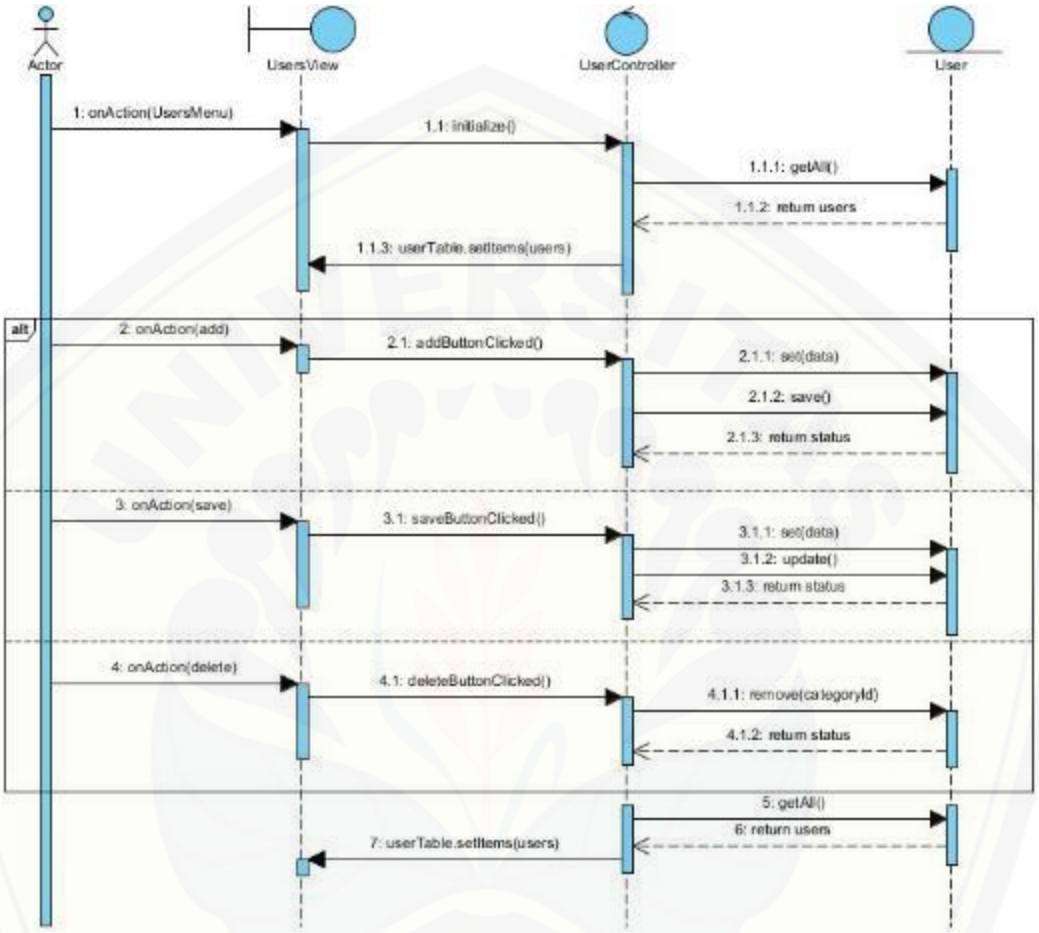
Sequence ini diakses oleh manajer dan pegawai digunakan untuk mengelola data konsumsi kategori trafo.



Gambar C. 1 Sequence Diagram Mengelola Data Kategori Trafo

C.2. *Sequence Diagram* Mengelola Data Pengguna

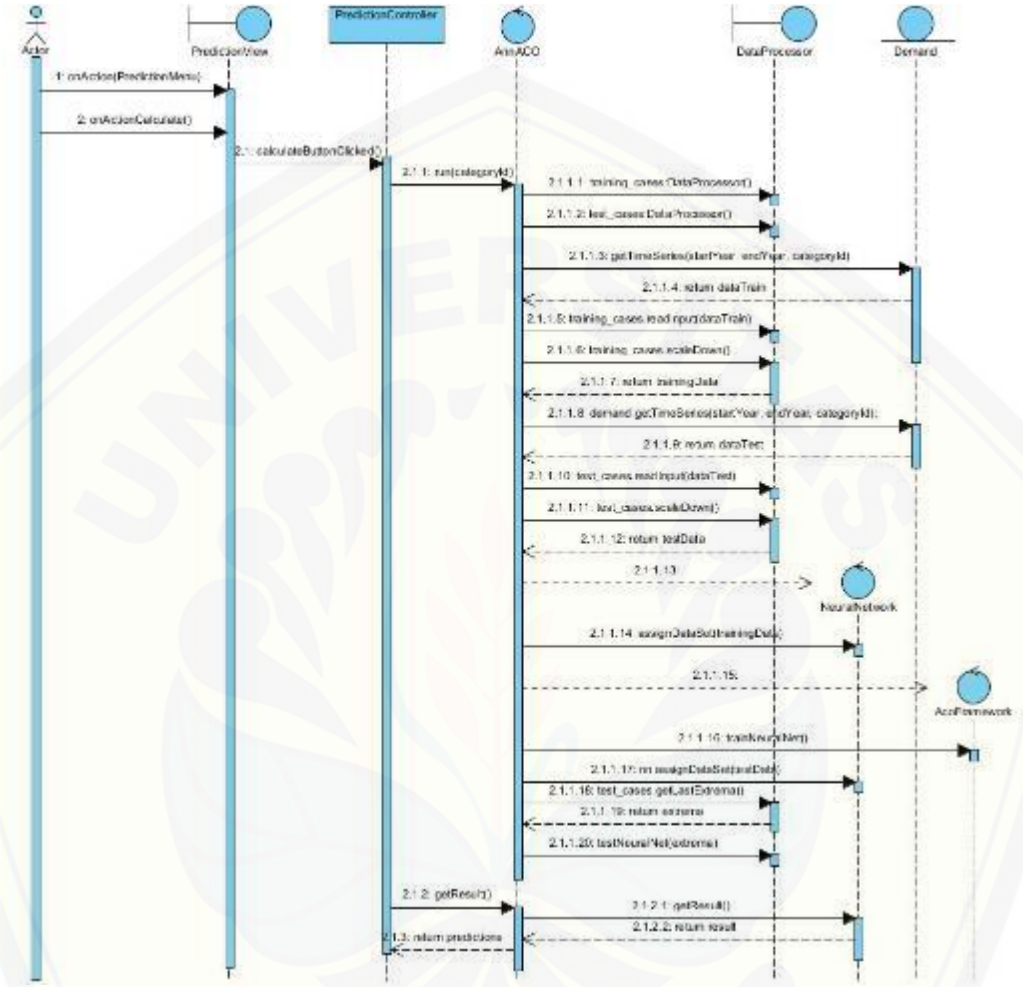
Sequence ini diakses oleh manajer digunakan untuk mengelola data pengguna.



Gambar C. 2 Sequence Diagram Mengelola Data Pengguna

C.3. *Sequence Diagram* Kalkulasi Prediksi Kebutuhan Trafo

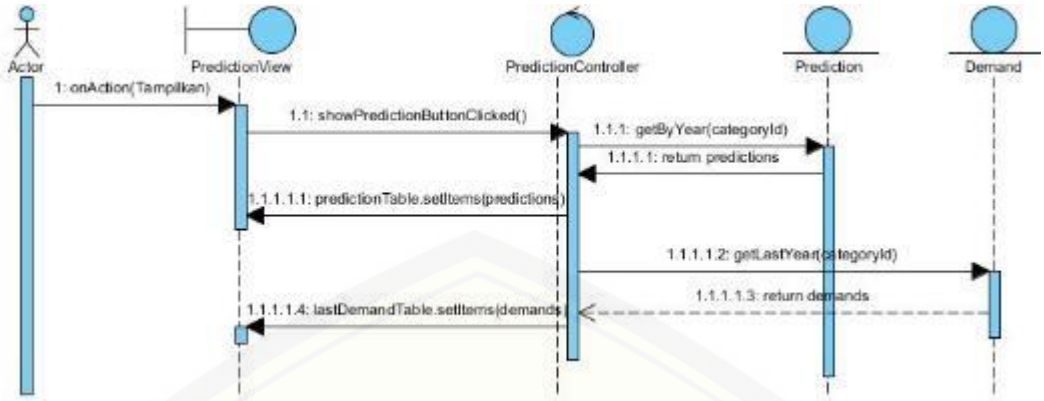
Sequence ini diakses oleh manajer digunakan untuk menjalankan kalkulasi prediksi kebutuhan trafo.



Gambar C. 3 Sequence Diagram Kalkulasi Prediksi Kebutuhan Trafo

C.4. *Sequence Diagram* Melihat Hasil Prediksi Kebutuhan Trafo

Sequence ini diakses oleh manajer digunakan untuk melihat hasil kalkulasi prediksi kebutuhan trafo.



Gambar C. 4 Sequence Diagram Melihat Hasil Prediksi Kebutuhan Trafo

LAMPIRAN D Kode Program

D.1. Kode Login

Penulisan kode login dalam sistem optimasi persediaan trafo terletak di class *LoginController* pada *controller*. Penulisan kode login dapat dilihat pada tabel berikut ini:

Tabel D. 1 Kode Login

Class LoginController pada *controller*

```
package app.controller;

import app.model.User;
import app.utility.Route;
import app.utility.UserSession;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.SplitPane;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import java.io.File;
```

```
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class LoginController implements Initializable {

    @FXML
    private ImageView loginBackground;

    @FXML
    private SplitPane splitPane;

    @FXML
    private TextField username;

    @FXML
    private PasswordField password;

    @FXML
    public Button loginButton;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        File file = new File("src/assets/login_bg.jpg");
        Image image = new Image(file.toURI().toString());
        loginBackground.setImage(image);
        splitPane.lookupAll(".split-pane-divider").stream()
                .forEach(div ->  div.setMouseTransparent(true) );
    }

    public void signinButtonClicked(){
        User user = new User();
        user.setUsername(username.getText());
        user.setPassword(password.getText());
        // Cek Status Login
        if(user.login() != null) {
            // Session
            UserSession.getInstace(user.login());
            // Tutup Halaman Login
            Stage loginStage = (Stage) loginButton.getScene().getWindow();
```

```
            loginStage.close();
            try {
                URL url = new File("src/app/view/DemandView.fxml").toURL();
                Parent dashboard = FXMLLoader.load(url);
                // Tampilkan Halaman Dashboard
                Stage mainStage = new Stage();
                mainStage.setTitle("Si Trafo - Permintaan");
                mainStage.setScene(new Scene(dashboard));
                mainStage.show();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("Failed to login");
        }
    }
}
```

D.2. Kode Pengguna

Penulisan kode pengguna dalam sistem optimasi persediaan trafo terletak di *class User* pada *model*, dan *UserController* pada *controller*. Penulisan kode pengguna dapat dilihat pada tabel berikut ini:

Tabel D. 2 Kode Pengguna

Class User pada model

```
package app.model;

import app.connection.DBConnect;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;

public class User {
```

```
    private int id;
    private String fullName;
    private String username;
    private String password;
    private String level;
    private Date created_at;
    private Date updated_at;

    private Connection connection;
    private String table = "users";
    ObservableList<User> users;

    public User() {connection = DBConnect.connect(); }

    public User(String fullName, String username, String level) {
        this.fullName = fullName;
        this.username = username;
        this.level = level;
    }

    public ObservableList<User> getAll() {
        try{
            //SQL FOR SELECTING ALL OF CUSTOMER
            String SQL = "SELECT * from " + table;
            users = FXCollections.observableArrayList();
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                users.add(new      User(rs.getString(2),      rs.getString(3),
rs.getString(4)));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return users;
    }

    public User login(){
        User user = null;
```

```
        try{
            //SQL FOR SELECTING ALL OF CUSTOMER
            String SQL = "SELECT * from " + table + " WHERE username = '" +
username + "' AND password = '" + password + "'";
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                user    =    new    User(rs.getString(2),    rs.getString(3),
rs.getString(4));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return user;
    }

    public Boolean save() {
        try{
            String SQL = "INSERT INTO " + table
                    + " VALUES ("
                    + "null,"
                    + "'" + fullName + "',"
                    + "'" + username + "',"
                    + "'" + level + "',"
                    + "'" + password + "',"
                    + created_at + ","
                    + updated_at + ")";
            System.out.println("Execute: " + SQL);
            Statement st = connection.createStatement();
            st.execute(SQL);
            connection.close();
            return true;
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error when inserting Data User");
            return false;
        }
    }
```

```
    private String securePassword(String password)
    {
        String securedPassword = null;
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(password.getBytes());
            byte[] bytes = md.digest();
            StringBuilder sb = new StringBuilder();
            for(int i=0; i< bytes.length ;i++)
            {
                sb.append(Integer.toString((bytes[i]   &   0xff)   +   0x100,
16).substring(1));
            }
            securedPassword = sb.toString();
        }
        catch (NoSuchAlgorithmException e)
        {
            e.printStackTrace();
        }
        return securedPassword;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public String getUsername() {
        return username;
```

```
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = securePassword(password);
    }

    public String getLevel() {
        return level;
    }

    public void setLevel(int level) {
        if(level == 0)
            this.level = "Manajer";
        else
            this.level = "Pegawai";
    }

    public Date getCreated_at() {
        return created_at;
    }

    public void setCreated_at(Date created_at) {
        this.created_at = created_at;
    }

    public Date getUpdated_at() {
        return updated_at;
    }

    public void setUpdated_at(Date updated_at) {
        this.updated_at = updated_at;
    }
}
```

Class UserController pada *controller*

```
package app.controller;

import app.model.User;
import app.utility.Route;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import java.net.URL;
import java.util.ResourceBundle;

public class UserController implements Initializable, Route {
    private User user;

    @FXML
    private Button demandMenu;

    @FXML
    private Button categoryMenu;

    @FXML
    private Button predictionMenu;

    @FXML
    private Button usersMenu;

    @FXML
    private TableView usersTable;

    @FXML
    private TextField fullNameInput;

    @FXML
    private TextField usernameInput;

    @FXML
    private ComboBox level;

    @FXML
    private PasswordField passwordInput;
```

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    user = new User();

    TableColumn<User, String> fullNameColumn = new TableColumn<>("Nama Lengkap");
    fullNameColumn.setMinWidth(200);
    fullNameColumn.setCellValueFactory(new PropertyValueFactory<>("fullName"));

    TableColumn<User, String> usernameColumn = new TableColumn<>("Username");
    usernameColumn.setCellValueFactory(new PropertyValueFactory<>("username"));

    TableColumn<User, String> levelColumn = new TableColumn<>("Level");
    levelColumn.setCellValueFactory(new PropertyValueFactory<>("level"));

    usersTable.getColumns().addAll(fullNameColumn, usernameColumn, levelColumn);
    usersTable.setItems(user.getAll());

    level.getItems().addAll("Manajer", "Pegawai");
    level.setValue("Pegawai");
}

public void addButtonClicked(){
    user = new User();
    user.setFullName(fullNameInput.getText());
    user.setUsername(usernameInput.getText());
    user.setLevel(level.getSelectionModel().getSelectedIndex());
    user.setPassword(passwordInput.getText());

    if(user.save()){
        usersTable.getItems().add(user);
    }
}

public void deleteButtonClicked(){
    ObservableList<User> userSelected, allUsers;
    allUsers = usersTable.getItems();
    userSelected = usersTable.getSelectionModel().getSelectedItems();

    userSelected.forEach(allUsers::remove);
```

```
        // TODO Remove From Database
    }

    @Override
    public void demandMenuClicked() {
        MenuController.demandMenuClicked(demandMenu);
    }

    @Override
    public void categoryMenuClicked() {
        MenuController.categoryMenuClicked(categoryMenu);
    }

    @Override
    public void predictionMenuClicked() {
        MenuController.predictionMenuClicked(predictionMenu);
    }

    @Override
    public void usersMenuClicked() {
        MenuController.usersMenuClicked(usersMenu);
    }

    @Override
    public void logoutClicked() {
        MenuController.logoutClicked();
    }
}
```

D.3. Kode Demand

Penulisan kode demand dalam sistem optimasi persediaan trafo terletak di *class Demand* pada *model*, dan *DemandController* pada *controller*. Penulisan kode demand dapat dilihat pada tabel berikut ini:

Tabel D. 3 Kode *Demand*

Class Demand pada *model*

```
package app.model;

import app.connection.DBConnect;
```

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.*;
import java.util.Date;

public class Demand {
    private int id;
    private int categoryId;
    private String categoryName;
    private int month;
    private String monthName;
    private int year;
    private int amount;
    private Date created_at;
    private Date updated_at;

    private Connection connection;
    private String table = "demands";
    ObservableList<Demand> demands;

    public Demand() {
        connection = DBConnect.connect();
    }

    public Demand(int id, String categoryName, String monthName, int year, int
amount, Date created_at, Date updated_at) {
        this.id = id;
        this.categoryName = categoryName;
        this.monthName = monthName;
        this.year = year;
        this.amount = amount;
        this.created_at = created_at;
        this.updated_at = updated_at;
    }

    public ObservableList<Demand> getAll(int categoryId) {
        try{
            //SQL FOR SELECTING ALL OF DEMAND
```

```
            String SQL = "SELECT t.*, c.type, c.volt from " + table + " t JOIN
categories c ON t.category_id = c.id WHERE category_id = " + categoryId;
            demands = FXCollections.observableArrayList();
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                demands.add(new Demand(rs.getInt(1), rs.getString(8) + " " +
rs.getString(9)   +   "   MW",   getMonthName(rs.getInt(3)),   rs.getInt(4),
rs.getInt(5), rs.getDate(6), rs.getDate(7)));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return demands;
    }

    public ObservableList<Demand> getLastYear(int categoryId) {
        try{
            //SQL FOR SELECTING ALL OF DEMAND
            String SQL = "SELECT t.*, c.type, c.volt from " + table + " t JOIN
categories c ON t.category_id = c.id WHERE category_id = " + categoryId + " AND
year = YEAR(DATE_SUB(CURDATE(), INTERVAL 1 YEAR))";
            demands = FXCollections.observableArrayList();
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                demands.add(new Demand(rs.getInt(1), rs.getString(8) + " " +
rs.getString(9)   +   "   MW",   getMonthName(rs.getInt(3)),   rs.getInt(4),
rs.getInt(5), rs.getDate(6), rs.getDate(7)));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return demands;
    }

    public Boolean save() {
        try{
```

```
            String SQL = "INSERT INTO " + table
                    + " VALUES ("
                    + "null,"
                    + categoryId + ","
                    + month + ","
                    + year + ","
                    + amount + ","
                    + created_at + ","
                    + updated_at + ")";
            System.out.println("Execute: " + SQL);
            Statement st = connection.createStatement();
            st.execute(SQL);
            connection.close();
            return true;
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error when inserting Data");
            return false;
        }
    }

    private String getMonthName(int month) {
        String monthName;
        switch (month) {
            case 1:
                monthName = "Januari";
                break;
            case 2:
                monthName = "February";
                break;
            case 3:
                monthName = "Maret";
                break;
            case 4:
                monthName = "April";
                break;
            case 5:
                monthName = "Mei";
                break;
            case 6:
```

```
                monthName = "Juni";
                break;
            case 7:
                monthName = "Juli";
                break;
            case 8:
                monthName = "Agustus";
                break;
            case 9:
                monthName = "September";
                break;
            case 10:
                monthName = "Oktober";
                break;
            case 11:
                monthName = "November";
                break;
            case 12:
                monthName = "Desember";
                break;
            default:
                monthName = "";
                break;
        }
        return monthName;
    }

    public int[][] getTimeSeries(int startYear, int endYear, int categoryId){
        try{
            String SQL = "SELECT amount from " + table + " WHERE year BETWEEN
" + startYear + " AND " + endYear + " AND category_id = " + categoryId;
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            int size = 0;
            if(rs != null){
                rs.last();
                size = rs.getRow();
            }
            rs.first();
            int yearDistance = size/12;
            System.out.println("Year distance: " + yearDistance);
```

```
            int[][] data = new int[size][13];
            int pos = 0;
            for(int i = 0; i < yearDistance - 1; i++){
                for(int j = 0; j < 12; j++){
                    System.out.print(pos + 1);
                    System.out.print(" = ");
                    for(int k = 0; k < 13; k++){
                        System.out.print(rs.getInt(1) + " ");
                        data[pos][k] = rs.getInt(1);
                        rs.next();
                    }
                    for(int l = 0; l < 12; l++){
                        rs.previous();
                    }
                    System.out.println(" ");
                    pos++;
                }
            }
            return data;
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Error reading data from database");
            return null;
        }
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getCategoryId() {
        return categoryId;
    }

    public void setCategoryId(int categoryId) {
        this.categoryId = categoryId;
```

```
    }

    public int getMonth() {
        return month;
    }

    public void setMonth(int month) {
        this.month = month;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public int getAmount() {
        return amount;
    }

    public void setAmount(int amount) {
        this.amount = amount;
    }

    public Date getCreated_at() {
        return created_at;
    }

    public void setCreated_at(Date created_at) {
        this.created_at = created_at;
    }

    public Date getUpdated_at() {
        return updated_at;
    }

    public void setUpdated_at(Date updated_at) {
        this.updated_at = updated_at;
```

```
    }

    public String getCategoryName() {
        return categoryName;
    }

    public void setCategoryName(String categoryName) {
        this.categoryName = categoryName;
    }

    public String getMonthName() {
        return monthName;
    }

    public void setMonthName(String monthName) {
        this.monthName = monthName;
    }

    public boolean remove(int id) {
        try {
            PreparedStatement statement = connection.prepareStatement("DELETE
FROM "+ table +" WHERE id = ?");
            statement.setInt(1, id);
            statement.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public Boolean update() {
        try {
            String SQL = "UPDATE " + table
                    +" SET `month` = '" + month + "', "
                    + "`year` = '" + year + "', "
                    + "`amount` = '" + amount + "' "
                    + "WHERE `"+ table +"`.`id` = " + id;
            System.out.println("Execute: " + SQL);
            Statement st = connection.createStatement();
```

```
            st.execute(SQL);
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("Error when updating Data");
        }
        return false;
    }
}
```

Class DemandController pada *controller*

```
package app.controller;

import app.model.Category;
import app.model.Demand;
import app.model.User;
import app.utility.Route;
import app.utility.UserSession;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.HBox;

import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.ResourceBundle;

public class DemandController implements Initializable, Route {
    private Demand demand;
    private ObservableList<String> categoriesString;
    private Category category;

    @FXML
    private TableView demandTable;

    @FXML
```

```
    private ComboBox categoryCombobox;

    @FXML
    private ComboBox monthInput;

    @FXML
    private ComboBox yearInput;

    @FXML
    private TextField amountInput;

    @FXML
    private Button demandMenu;

    @FXML
    private Button categoryMenu;

    @FXML
    private Button predictionMenu;

    @FXML
    private Button usersMenu;

    @FXML
    private Button addButton;

    @FXML
    private Button saveButton;

    @FXML
    private HBox crudPanel;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        demand = new Demand();
        category = new Category();
        categoriesString = category.getComboboxString();
        categoryCombobox.setItems(categoriesString);

        TableColumn<Demand, String> idColumn = new TableColumn<>("ID");
```

```
        idColumn.setMinWidth(20);
        idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

        TableColumn<Demand,      String>      categoryColumn      =      new
TableColumn<>("Jenis");
        categoryColumn.setMinWidth(200);
        categoryColumn.setCellValueFactory(new
PropertyValueFactory<>("categoryName"));

        TableColumn<Demand, String> monthColumn = new TableColumn<>("Bulan");
        monthColumn.setCellValueFactory(new
PropertyValueFactory<>("monthName"));

        TableColumn<Demand, String> yearColumn = new TableColumn<>("Tahun");
        yearColumn.setCellValueFactory(new PropertyValueFactory<>("year"));

        TableColumn<Demand, String> amountColumn = new TableColumn<>("Jumlah");
        amountColumn.setCellValueFactory(new
PropertyValueFactory<>("amount"));

        demandTable.getColumns().addAll(idColumn, categoryColumn, monthColumn,
yearColumn, amountColumn);

        monthInput.getItems().setAll("January", "February", "Maret", "April",
"Mei", "Juni", "Juli", "Agustus", "September",
                "Oktober", "November", "Desember");

        ArrayList<Integer> year = new ArrayList<>();
        for (int i = Calendar.getInstance().get(Calendar.YEAR); i >= 2015; i--
){
            year.add(i);
        }
        yearInput.getItems().setAll(year);

        crudPanel.setVisible(false);
    }

    public void getDemandsButtonClicked() {
        ArrayList<Category> categories = category.getCombobox();
        for (Category cat : categories) {
```

```
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                demandTable.setItems(demand.getAll(cat.getId()));
            }
        }
        if(UserSession.getLevel().equals("Pegawai")) {
            crudPanel.setVisible(true);
        }
    }

    public void addButtonClicked() {
        Demand demand = new Demand();

        ArrayList<Category> categories = category.getCombobox();
        for (Category cat : categories) {
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                demand.setCategoryId(cat.getId());
            }
        }

        demand.setMonth(monthInput.getSelectionModel().getSelectedIndex()    +
1);
        demand.setYear(Integer.parseInt(yearInput.getValue().toString()));
        demand.setAmount(Integer.parseInt(amountInput.getText()));
        demand.setMonthName(monthInput.getValue().toString());
        demand.setCategoryName(categoryCombobox.getValue().toString());
        if(demand.save()){
            demandTable.getItems().clear();
            getDemandsButtonClicked();
            monthInput.getSelectionModel().clearSelection();
            monthInput.setPromptText("Pilih Bulan");
            yearInput.getSelectionModel().clearSelection();
            yearInput.setPromptText("Pilih Tahun");
            amountInput.setText("");
        } else {
            System.out.println("Kesalahan.  Terjadi  Kesalahan  saat  menginput
data");
        }
    }

    public void editButtonClicked(){
```

```
        ObservableList<Demand>                demandSelected                =
demandTable.getSelectionModel().getSelectedItems();

amountInput.setText(String.valueOf(demandSelected.get(0).getAmount()));

        addButton.setVisible(false);
        saveButton.setVisible(true);
    }

    public void saveButtonClicked(){
        ObservableList<Demand>                demandSelected                =
demandTable.getSelectionModel().getSelectedItems();
        Demand dem = new Demand();

        dem.setId(demandSelected.get(0).getId());
        dem.setMonth(monthInput.getSelectionModel().getSelectedIndex() + 1);
        dem.setYear(Integer.parseInt(yearInput.getValue().toString()));
        dem.setAmount(Integer.parseInt(amountInput.getText()));
        if(dem.update()){
            demandTable.getItems().clear();
            getDemandsButtonClicked();
            monthInput.getSelectionModel().clearSelection();
            monthInput.setPromptText("Pilih Bulan");
            yearInput.getSelectionModel().clearSelection();
            yearInput.setPromptText("Pilih Tahun");
            amountInput.setText("");
        }
        addButton.setVisible(true);
        saveButton.setVisible(false);
    }

    public void deleteButtonClicked(){
        ObservableList<Demand> demandSelected, allDemands;
        allDemands = demandTable.getItems();
        demandSelected = demandTable.getSelectionModel().getSelectedItems();

        if(demand.remove(demandSelected.get(0).getId()))
            demandSelected.forEach(allDemands::remove);
        else
            System.out.println("Error while deleting data");
```

```
    }

    @Override
    public void demandMenuClicked() {
        MenuController.demandMenuClicked(demandMenu);
    }

    @Override
    public void categoryMenuClicked() {
        MenuController.categoryMenuClicked(categoryMenu);
    }

    @Override
    public void predictionMenuClicked() {
        MenuController.predictionMenuClicked(predictionMenu);
    }

    @Override
    public void usersMenuClicked() {
        MenuController.usersMenuClicked(usersMenu);
    }

    @Override
    public void logoutClicked() {
        MenuController.logoutClicked();
    }
}
```

D.4. Kode Kategori

Penulisan kode kategori dalam sistem optimasi persediaan trafo terletak di *Category* pada *model*, dan *CategoryController* pada *controller*. Penulisan kode kategori dapat dilihat pada tabel berikut ini:

Tabel D. 4 Kode Kategori

Class Category pada *model*

```
package app.model;

import app.connection.DBConnect;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.*;
import java.util.ArrayList;
import java.util.Date;

public class Category {
    private int id;
    private String idString;
    private String type;
    private int volt;
    private String typeVolt;
    private Date created_at;
    private Date updated_at;

    private Connection connection;
    private String table = "categories";
    ObservableList<Category> categories;

    public Category() {connection = DBConnect.connect(); }

    public Category(int id, String type, int volt, Date created_at, Date
updated_at) {
        this.id = id;
        this.type = type;
        this.volt = volt;
        this.created_at = created_at;
        this.updated_at = updated_at;
    }

    public Category(String id, String typeVolt) {
        this.idString = id;
        this.typeVolt = typeVolt;
```

```
    }

    public Category(int id, String typeVolt) {
        this.id = id;
        this.typeVolt = typeVolt;
    }

    public ObservableList<Category> getAll() {
        try{
            //SQL FOR SELECTING ALL OF CATEGORIES
            String SQL = "SELECT * from " + table;
            categories = FXCollections.observableArrayList();
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                categories.add(new   Category(rs.getInt(1),   rs.getString(2),
rs.getInt(3), rs.getDate(4), rs.getDate(5)));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return categories;
    }


    public ObservableList<String> getComboboxString() {
        ObservableList<String> categoriesString;
        categoriesString = FXCollections.observableArrayList();
        try{
            //SQL FOR SELECTING ALL OF CATEGORIES
            String SQL = "SELECT id, type, volt from " + table;
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                categoriesString.add(rs.getString(2) + " " + rs.getInt(3));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
```

```
        }
        return categoriesString;
    }

    public ArrayList<Category> getCombobox() {
        ArrayList<Category> categories = new ArrayList<>();
        try{
            //SQL FOR SELECTING ALL OF CATEGORIES
            String SQL = "SELECT id, type, volt from " + table;
            //ResultSet
            ResultSet rs = connection.createStatement().executeQuery(SQL);
            while(rs.next()){
                categories.add(new Category(rs.getInt(1), rs.getString(2) + "
" + rs.getInt(3)));
            }
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
        return categories;
    }

    public Boolean save() {
        try{
             String SQL = "INSERT INTO " + table
                    + " VALUES ("
                    + "null,"
                    + "'" + type + "',"
                    + volt + ","
                    + created_at + ","
                    + updated_at + ")";
            System.out.println("Execute: " + SQL);
            Statement st = connection.createStatement();
            st.execute(SQL);
            return true;
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error when inserting Data");
            return false;
        }
```

```
    }

    public Boolean update() {
        try {
            String SQL = "UPDATE " + table
                    +" SET `type` = '"+ type+"', "
                    + "`volt` = '" + volt + "' "
                    + "WHERE `"+ table +"`.`id` = " + id;
            System.out.println("Execute: " + SQL);
            Statement st = connection.createStatement();
            st.execute(SQL);
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("Error when updating Data");
        }
        return false;
    }

    public Boolean remove(int id){
        try {
            PreparedStatement statement = connection.prepareStatement("DELETE
FROM "+ table +" WHERE id = ?");
            statement.setInt(1, id);
            statement.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public int getVolt() {
        return volt;
    }

    public void setVolt(int volt) {
        this.volt = volt;
    }

    public Date getCreated_at() {
        return created_at;
    }

    public void setCreated_at(Date created_at) {
        this.created_at = created_at;
    }

    public Date getUpdated_at() {
        return updated_at;
    }

    public void setUpdated_at(Date updated_at) {
        this.updated_at = updated_at;
    }

    public String getIdString() {
        return idString;
    }

    public String getTypeVolt() {
        return typeVolt;
    }
}
```

Class CategoryController pada *controller*

```
package app.controller;

import app.model.Category;
import app.utility.Route;
import app.utility.UserSession;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.HBox;

import java.net.URL;
import java.util.ResourceBundle;

public class CategoryController implements Initializable, Route {

    Category category;

    @FXML
    private TextField typeInput;

    @FXML
    private TextField voltInput;

    @FXML
    private TableView categoryTable;

    @FXML
    private Button demandMenu;

    @FXML
    private Button categoryMenu;
```

```
@FXML
private Button predictionMenu;

@FXML
private Button usersMenu;

@FXML
private Button addButton;

@FXML
private Button saveButton;

@FXML
private HBox crudPanel;

@Override
public void initialize(URL location, ResourceBundle resources) {
    category = new Category();

    TableColumn<Category, String> idColumn = new TableColumn<>("ID");
    idColumn.setMinWidth(20);
    idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

    TableColumn<Category, String> typeColumn = new TableColumn<>("Jenis");
    typeColumn.setMinWidth(200);
    typeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));

    TableColumn<Category, String> voltColumn = new TableColumn<>("Volt");
    voltColumn.setCellValueFactory(new PropertyValueFactory<>("volt"));

    categoryTable.getColumns().addAll(idColumn, typeColumn, voltColumn);
    categoryTable.setItems(category.getAll());

    if(UserSession.getLevel().equals("Pegawai")){
        crudPanel.setVisible(false);
    }
    saveButton.setVisible(false);
}

public void addButtonClicked(){
```

```
        Category category = new Category();
        category.setType(typeInput.getText());
        category.setVolt(Integer.parseInt(voltInput.getText()));
        if(category.save()) {
            categoryTable.getItems().clear();
            categoryTable.setItems(category.getAll());
            typeInput.setText("");
            voltInput.setText("");
        }
    }

    public void editButtonClicked(){
        ObservableList<Category>              categorySelected              =
categoryTable.getSelectionModel().getSelectedItems();

        typeInput.setText(categorySelected.get(0).getType());
        voltInput.setText(String.valueOf(categorySelected.get(0).getVolt()));
        addButton.setVisible(false);
        saveButton.setVisible(true);
    }

    public void saveButtonClicked(){
        ObservableList<Category>              categorySelected              =
categoryTable.getSelectionModel().getSelectedItems();
        Category cat = new Category();
        cat.setId(categorySelected.get(0).getId());
        cat.setType(typeInput.getText());
        cat.setVolt(Integer.parseInt(voltInput.getText()));
        if(cat.update()){
            categoryTable.getItems().clear();
            categoryTable.setItems(category.getAll());
            typeInput.setText("");
            voltInput.setText("");
        }
        addButton.setVisible(true);
        saveButton.setVisible(false);
    }

    public void deleteButtonClicked(){
        ObservableList<Category> categorySelected, allCategories;
```

```
        allCategories = categoryTable.getItems();
        categorySelected                                                    =
categoryTable.getSelectionModel().getSelectedItems();

        if(category.remove(categorySelected.get(0).getId()))
            categorySelected.forEach(allCategories::remove);
        else
            System.out.println("Error while deleting data");
    }

    @Override
    public void demandMenuClicked() {
        MenuController.demandMenuClicked(demandMenu);
    }

    @Override
    public void categoryMenuClicked() {
        MenuController.categoryMenuClicked(categoryMenu);
    }

    @Override
    public void predictionMenuClicked() {
        MenuController.predictionMenuClicked(predictionMenu);
    }

    @Override
    public void usersMenuClicked() {
        MenuController.usersMenuClicked(usersMenu);
    }

    @Override
    public void logoutClicked() {
        MenuController.logoutClicked();
    }
}
```

D.5. Kode Prediksi

Penulisan kode prediksi dalam sistem optimasi persediaan trafo terletak di *class* AnnACO, ACOFramework, DataProcessor, NeuralNetwork dan Neuron pada *controller*. Penulisan kode prediksi dapat dilihat pada tabel berikut ini:

Tabel D. 5 Kode Prediksi

Class PredictionController pada *controller*

```
package app.controller;

import app.calculation.AnnACO;
import app.model.Category;
import app.model.Demand;
import app.model.Prediction;
import app.utility.Route;
import app.utility.UserSession;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.VBox;

import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

public class PredictionController implements Initializable, Route {
    private ObservableList<String> categoriesString;
    private Demand demand;
    private Category category;
    private AnnACO annACO;
    private Prediction prediction;

    @FXML
    private Button demandMenu;

    @FXML
    private Button categoryMenu;
```

```
    @FXML
    private Button predictionMenu;

    @FXML
    private Button usersMenu;

    @FXML
    private ComboBox categoryCombobox;

    @FXML
    private TableView predictionTable;

    @FXML
    private TableView lastDemandTable;

    @FXML
    private Label lastIteration;

    @FXML
    private Label finalFitness;

    @FXML
    private Label predictionYear;

    @FXML
    private Button calculateButton;

    @FXML
    private Button showPredictionButton;

    @FXML
    private Button predictionButton;

    @FXML
    private Button saveButton;

    @FXML
    private VBox managerPanel;

    @Override
```

```
    public void initialize(URL location, ResourceBundle resources) {
        demand = new Demand();
        category = new Category();
        annACO = new AnnACO();
        categoriesString = category.getComboboxString();
        categoryCombobox.setItems(categoriesString);

        TableColumn<Demand, String> demandMonthColumn = new
TableColumn<>("Bulan");
        demandMonthColumn.setCellValueFactory(new
PropertyValueFactory<>("monthName"));

        TableColumn<Demand, String> demandAmountColumn = new
TableColumn<>("Jumlah");
        demandAmountColumn.setCellValueFactory(new
PropertyValueFactory<>("amount"));

        lastDemandTable.getColumns().addAll(demandMonthColumn,
demandAmountColumn);

        TableColumn<Prediction, String> monthColumn = new
TableColumn<>("Bulan");
        monthColumn.setCellValueFactory(new
PropertyValueFactory<>("monthName"));

        TableColumn<Prediction, String> amountColumn = new
TableColumn<>("Jumlah");
        amountColumn.setCellValueFactory(new
PropertyValueFactory<>("amountPrediction"));

        predictionTable.getColumns().addAll(monthColumn, amountColumn);
        if (UserSession.getLevel().equals("Pegawai")) {
            calculateButton.setVisible(false);
            managerPanel.setVisible(false);
            showPredictionButton.setVisible(true);
            predictionYear.setText("2019");
        } else {
            showPredictionButton.setVisible(false);
        }
        predictionButton.setVisible(false);
```

```
        saveButton.setVisible(false);
    }

    public void calculateButtonClicked(){
        ArrayList<Category> categories = category.getCombobox();

        for (Category cat : categories) {
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                lastDemandTable.setItems(demand.getLastYear(cat.getId()));
                annACO.run(cat.getId());
                predictionTable.setItems(annACO.getResult(false));

lastIteration.setText(String.valueOf(annACO.getLastIteration()));

finalFitness.setText(String.valueOf(annACO.getFinalFitness()));
                predictionButton.setVisible(true);
                predictionYear.setText("2018");
            }
        }
    }

    public void predictionButtonClicked(){
        ArrayList<Category> categories = category.getCombobox();

        for (Category cat : categories) {
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                annACO.getPrediction(cat.getId());
                predictionTable.getItems().clear();
                predictionTable.setItems(annACO.getResult(true));
                predictionYear.setText("2019");
                saveButton.setVisible(true);
            }
        }
    }

    public void saveButtonClicked(){
        // Simpan Solusi Bobot
        annACO.saveSolution();
        ArrayList<Category> categories = category.getCombobox();
```

```
        for (Category cat : categories) {
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                annACO.saveResult(cat.getId());
            }
        }
    }

    public void showPredictionButtonClicked(){
        ArrayList<Category> categories = category.getCombobox();

        for (Category cat : categories) {
            if (cat.getTypeVolt().equals(categoryCombobox.getValue())) {
                prediction = new Prediction();
                lastDemandTable.setItems(demand.getLastYear(cat.getId()));
                predictionTable.setItems(prediction.getByYear(cat.getId(),
2019));
            }
        }

    }

    @Override
    public void demandMenuClicked() {
        MenuController.demandMenuClicked(demandMenu);
    }

    @Override
    public void categoryMenuClicked() {
        MenuController.categoryMenuClicked(categoryMenu);
    }

    @Override
    public void predictionMenuClicked() {
        MenuController.predictionMenuClicked(predictionMenu);
    }

    @Override
    public void usersMenuClicked() {
        MenuController.usersMenuClicked(usersMenu);
    }
```

```
    @Override
    public void logoutClicked() {
        MenuController.logoutClicked();
    }
}
```

Class AnnACO pada *controller*

```
package app.calculation;

import app.connection.DBConnect;
import app.model.Demand;
import app.model.Prediction;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.Connection;
import java.sql.Statement;

public class AnnACO {

    static NeuralNetwork nn;
    static AcoFramework AF;
    int startYear;
    int endYear;

    public void run(int categoryId) {
        double[][] trainingData, testData;
        startYear = 2012;
        endYear = 2018;
        int numcases = (endYear - startYear)*12;
        int numInputs = 12;
        int numOutputs = 1;
        int repoSize = 20; //size of the repository

        // read training and test data.
        DataProcessor training_cases = new DataProcessor(numcases, numInputs,
numOutputs);
```

```
        DataProcessor test_cases = new DataProcessor(numcases, numInputs,
numOutputs);

        Demand demand = new Demand();
        int[][] dataTrain = demand.getTimeSeries(startYear, endYear,
categoryId);
        training_cases.readInput(dataTrain);
        int[][] dataTest = demand.getTimeSeries(startYear, endYear,
categoryId);
        test_cases.readInput(dataTest);

        // Scale down training and test data.
        trainingData = training_cases.scaleDown();
        testData = test_cases.scaleDown();

        System.out.println("Scaled down training data:\n");
        for (int i = 0; i < numcases; i++) {
            for (int j = 0; j < numInputs + numOutputs; j++)
                System.out.print(trainingData[i][j] + " ");
            System.out.println("");
        }
        System.out.println("");
        System.out.println("End: Test data:");

        int[] numPerLayer = new int[3];
        numPerLayer[0] = numInputs;
        numPerLayer[1] = numInputs;
        numPerLayer[2] = numOutputs;
        nn = new NeuralNetwork(3, numInputs, numPerLayer);
        nn.assignDataSet(trainingData);
        AF = new AcoFramework(nn, repoSize);
        if (AF.trainNeuralNet()) {
            System.out.println("******Test Output*****");
            nn.assignDataSet(testData);
            double[] extrema = test_cases.getLastExtrema();
            AF.testNeuralNet(extrema);
        } else {
            System.exit(0);
        }
    }
```

```
    public ObservableList<Prediction> getResult(Boolean prediction){
        double[] result = nn.getResult();
        ObservableList<Prediction>                predictions                =
FXCollections.observableArrayList();

        if(!prediction) {
            for (int i = 0; i < 12; i++) {
                predictions.add(new Prediction(Prediction.getMonthName(i + 1),
result[i + ((endYear - startYear) * 12) - 12]));
            }
        } else {
            for (int i = 0; i < 12; i++) {
                predictions.add(new Prediction(Prediction.getMonthName(i + 1),
result[i]));
            }
        }

        return predictions;
    }

    public void saveSolution() {
        Connection connection = DBConnect.connect();
        double[] solution = AF.getSolution();
        try{
            String SQL = "TRUNCATE `sitrafo`.`bobot`";
            Statement st = connection.createStatement();
            st.execute(SQL);
            for(int i = 0; i < solution.length; i++) {
                SQL = "INSERT INTO bobot VALUES (NULL, " + solution[i] + ")";
                st = connection.createStatement();
                st.execute(SQL);
            }
            connection.close();
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error when inserting Data");
        }
    }
```

```
    public void saveResult(int categoryId){
        Connection connection = DBConnect.connect();
        double[] result = nn.getResult();
        try{
            String SQL = "TRUNCATE `predictions`";
            Statement st = connection.createStatement();
            st.execute(SQL);
            for(int i = 0; i < result.length; i++) {
                SQL = "INSERT INTO predictions " +
                        "VALUES (NULL, " +
                        "'"+categoryId+"', " +
                        "'"+(i+1)+"', " +
                        "'2019', " +
                        "'"+result[i]+"', " +
                        "NULL, NULL);";
                st = connection.createStatement();
                st.execute(SQL);
            }
            connection.close();
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Error when inserting Data");
        }
    }

    public Boolean getPrediction(int categoryId){
        System.out.println("Kategori ID = " + categoryId);
        double[][] sampleData;
        int numcases = 12;
        int numInputs = 12;
        int numOutputs = 1;

        DataProcessor sample_cases = new DataProcessor(numcases, numInputs,
numOutputs);
        Demand demand = new Demand();
        int[][] dataSample = demand.getTimeSeries(2018, 2019, categoryId);

        sample_cases.readInput(dataSample);
        sampleData = sample_cases.scaleDown();
        nn.assignDataSet(sampleData);
```

```
        double[] extrema = sample_cases.getLastExtrema();
        AF.testNeuralNet(extrema);

        return true;
    }

    public int getLastIteration(){
        return AF.getLastIteration();
    }

    public double getFinalFitness(){
        return AF.getFinalFitness();
    }
}
```

class ACOFramework pada *controller*

```
package app.calculation;

import org.apache.commons.math3.random.RandomDataImpl;

import java.util.Random;

public class AcoFramework {
    double[][] archive; /*Archive of NN weights*/
    int numLayers = 3; /*number of layers in the Neural Network*/
    /*array containing the number of nodes in each layer*/
    int[] nodesPerLayer;
    int numIns; /*number of inputs*/
    int numOuts; /* number of outputs*/
    int archiveSize; /*number of sets of NN weigthts in the archive*/
    int numWeights = 0; /*total number of weights in the neural network*/
    /*array to store the fitness value of each NN weight set*/
    double[] fitness;
    double[] solWeights; /*weight for each solution */
    double sumSolWeights = 0; /*sum of all solution weights*/
    double[] solution;
    NeuralNetwork nn; /*Neural network to be trained*/
    double epsilon = .85; /*affects pheromone evaporation rate*/
    double q = .08;
```

```
    Random rand;
    RandomDataImpl grand;
    double test_error = -1;
    double constant_sd = 0.1;
    int maxIterations = 1000000;
    int lastIteration = 0;
    double errorCriteria = 0.09;

    /*      *Constructor that creates the ACO framework.
     * *Takes a neural network and the size of the archive as parameters.     */
    AcoFramework(NeuralNetwork neuralNet, int archive_Size) {
        nn = neuralNet;
        archiveSize = archive_Size;
        nodesPerLayer = nn.nodesPerLayer;
        numIns = nodesPerLayer[0];
        numOuts = nodesPerLayer[nn.columns - 1];
        numWeights = nn.numWeights;
        initialize();
    }

    /*
    * Method to create a initialize the archive with random weights.
    */
    protected void initialize() {
        int i, j;
        archive = new double[archiveSize * 2][numWeights];
        fitness = new double[archiveSize * 2];
        solWeights = new double[archiveSize];
        rand = new Random();
        grand = new RandomDataImpl();

        /*          * fill archive with random values for weights          */
        for (i = 0; i < archiveSize * 2; i++)
            for (j = 0; j < (numWeights); j++) {
                archive[i][j] = rand.nextDouble() * 2 - 1;
            }
    }

    /*       * Method to compute the fitness of all the weight sets in the
archive.      */
```

```
public void computeFitness(int type, double[] lastExtrema) {
    if (type == 0) {
        for (int i = 0; i < archiveSize * 2; i++) {
            nn.setWeights(archive[i]);
            fitness[i] = nn.computeError(true, null);
        }
    }
    if (type == 1) {
        nn.setWeights(solution);
        test_error = nn.computeError(false, lastExtrema);
        System.out.println("Test error: " + test_error);
    }
}

/*
* Implementation of bubble sort algorithm to sort the archive according
* to the fitness of each solution. This method has a boolean parameter,
* which is set to true if the method is called to sort the just initialized
* array.
*/

public void sortArchive(boolean init) {
    int i, j;
    int n = archive.length;

    for (i = 0; i < n; i++)
        for (j = 0; j < n - 1; j++) {
            try {
                if (fitness[j] > fitness[j + 1]) {

                    double temp = fitness[j];
                    fitness[j] = fitness[j + 1];
                    fitness[j + 1] = temp;

                    double[] tempTrail = archive[j];
                    archive[j] = archive[j + 1];
                    archive[j + 1] = tempTrail;
                }
            } catch (Exception e) {
                System.out.println("error: " + j + " n " + n);
```

```
                    System.exit(0);
                }
            }
    }

    /*
    * Method to compute the weights for each set of weights in the archive
    */
    public void computeSolutionWeights() {
        sumSolWeights = 0;
        for (int i = 0; i < archiveSize; i++) {
            double exponent = (i * i) / (2 * q * q * archiveSize * archiveSize);
            solWeights[i] = (1 / (0.1 * Math.sqrt(2 * Math.PI))) *
Math.pow(Math.E, -exponent);
            sumSolWeights += solWeights[i];
        }
    }


    /*
    * Method to calculate the standard deviation of a particular weight of a
    *  particular weight set
    */
    protected double computeSD(int x, int l) {
        double sum = 0.0;
        for (int i = 0; i < archiveSize; i++) {
            sum += Math.abs(archive[i][x] - archive[l][x]) / (archiveSize - 1);
        }
        if (sum == 0) {
            return constant_sd;
        }
        return (epsilon * sum);
    }

    /*
    * select a Gaussian function that compose the Gaussian Kernel PDF
    */
    protected int selectPDF() {
        int i, l = 0;
        double temp = 0, prev_temp = 0;
```

```
        double r = rand.nextDouble();
        for (i = 0; i < archiveSize; i++) {
            temp += solWeights[i] / sumSolWeights;
            if (r < temp) {
                l = i;
                break;
            }
        }
        return l;
    }

    protected void generateBiasedWeights() {
        int i, j, pdf;
        double sigma; /*standard deviation*/
        double mu; /*mean*/

        pdf = 0;
        for (i = archiveSize; i < archiveSize * 2; i++) {
            pdf = selectPDF();
            for (j = 0; j < numWeights; j++) {
                sigma = computeSD(j, pdf);
                mu = archive[pdf][j];
                archive[i][j] = grand.nextGaussian(mu, sigma);
            }
        }
    }

    public boolean trainNeuralNet() {
        computeFitness(0, null);
        sortArchive(true);
        computeSolutionWeights();
        generateBiasedWeights();
        sortArchive(false);

        for (int j = 0; j < maxIterations; j++) {
            lastIteration = j;
            computeFitness(0, null);
            sortArchive(false);
            if (j % 1000 == 0) System.out.println("Iteration on " + (j/1000) +
"k: " + fitness[0]);
```

```
            if (fitness[0] < errorCriteria) {
                System.out.println("Solution found in iteration" + (j + 1));
                solution = archive[0];
                for (int i = 0; i < numWeights; i++)
System.out.println(archive[0][i]);
                return true;
            }
            computeSolutionWeights();
            generateBiasedWeights();
        }
        System.out.println("Network did not converge!");
        return false;
    }

    public void testNeuralNet(double[] lastExtrema) {
        computeFitness(1, lastExtrema);
    }

    public int getLastIteration() {
        return lastIteration;
    }

    public double getFinalFitness(){
        return fitness[0];
    }

    public double[] getSolution() {
        return solution;
    }

    public void setSolution(double[] solution) {
        this.solution = solution;
    }
}
```

Class DataProcessor pada *controller*

```
package app.calculation;

public class DataProcessor {
```

```
    int numInput;
    int numOutput;
    int numCases;
    double[][] trainingCases;
    double[] output;
    double[][] extrema;

    DataProcessor(int numCases, int numIns, int numOuts) {
        this.numCases = numCases;
        this.numInput = numIns;
        this.numOutput = numOuts;
        trainingCases = new double[numCases][numInput + numOutput];
        extrema = new double[numIns + numOuts][2];

        for (int i = 0; i < (numIns + numOuts); i++) {
            extrema[i][0] = 10000.0;
            extrema[i][1] = -10000.0;
        }
    }

    public void readInput(int[][] data) {
        try {
            for (int i = 0; i < numCases; i++) {
                for (int j = 0; j < numInput + numOutput; j++) {
                    trainingCases[i][j] = data[i][j];

                    if (trainingCases[i][j] < extrema[j][0])
                        extrema[j][0] = trainingCases[i][j];

                    if (trainingCases[i][j] > extrema[j][1])
                        extrema[j][1] = trainingCases[i][j];
                }
            }             //test whether both extrema are equal
            for (int i = 0; i < (numInput + numOutput); i++)
                if   (extrema[i][0]   ==   extrema[i][1])   extrema[i][1]   =
extrema[i][0] + 1;

            System.out.println("*************Extrema values*************");
```

```
            for (int i = 0; i < (numInput + numOutput); i++)
System.out.println(extrema[i][0] + " " + extrema[i][1]);
            System.out.println("*************End*************");

        } catch (Exception ex) {
            ex.printStackTrace();
            System.out.println("Error while reading file!");
        }

    }

    public void printOutput() {
        for (int i = 0; i < numCases; i++) {
            for (int j = 0; j < (numInput + numOutput); j++)
System.out.print(trainingCases[i][j] + " ");
            System.out.println("");
        }
    }

    /*
     * Scale Desired Output to 0..1
     */
    double[][] scaleDown() {
        double[][] scaledDownInput;
        int i, j;
        scaledDownInput = new double[numCases][numInput + numOutput];

        for (i = 0; i < numCases; i++)
            for (j = 0; j < numInput + numOutput; j++) {
                scaledDownInput[i][j] = .9 * (trainingCases[i][j] -
extrema[j][0]) / (extrema[j][1] - extrema[j][0]) + .05;
            }
        return scaledDownInput;
    }

    /*
     * Scale actual output to original range
     */
    double scaleOutput(double X, int which) {
        double range = extrema[which][1] - extrema[which][0];
```

```
        double scaleUp = ((X - .05) / .9) * range;
        return (extrema[which][0] + scaleUp);
    }

    public double[] getLastExtrema() {
        double[] lastExtrema = extrema[0];
        return lastExtrema;
    }
}
```

Class NeuralNetwork pada *controller*

```
package app.calculation;

public class NeuralNetwork {    /* 2-dimensional array of neurons that represent
the neural network*/
    Neuron[][] neuralNet;
    int columns;            /*number of columns including input layer*/
    int maxRows;            /*maximum number of rows in any column*/
    int[] nodesPerLayer;    /*number of nodes in each layer*/
    int numIns;
    int numOuts;
    int numWeights = 0;
    double[][] dataSet;
    double[] result;

    NeuralNetwork(int numLayers, int mRows, int[] numPerLayer) {
        this.maxRows = mRows;
        this.columns = numLayers;
        this.nodesPerLayer = numPerLayer;
        this.neuralNet = new Neuron[columns][maxRows];
        numIns = nodesPerLayer[0];
        numOuts = nodesPerLayer[numLayers - 1];

        for (int i = 1; i < columns; i++) numWeights = numWeights +
(nodesPerLayer[i] * nodesPerLayer[i - 1]);
        initNeurons();
    }

    protected void initNeurons() {
```

```
        int i, j;
        for (i = 0; i < numIns; i++) neuralNet[0][i] = new Neuron(0);

        for (i = 1; i < columns; i++)
            for (j = 0; j < nodesPerLayer[i]; j++) neuralNet[i][j] = new
Neuron(nodesPerLayer[i - 1]);
    }

    public void setWeights(double[] weights) {
        int x = 0;

        for (int i = 1; i < columns; i++)
            for (int j = 0; j < nodesPerLayer[i]; j++)
                for (int k = 0; k < neuralNet[i][j].numWeights; k++)
neuralNet[i][j].weights[k] = weights[x++];
    }

    public void assignDataSet(double[][] data_set) {
        this.dataSet = data_set;
    }

    protected double[] computeOutput(double[] inputs) {
        double sum;
        int i, j, k;
        double[] output = new double[nodesPerLayer[columns - 1]];

        /*Input Layer*/
        for (i = 0; i < nodesPerLayer[0]; i++) neuralNet[0][i].output =
inputs[i];

        /*Hidden Layers*/
        for (i = 1; i < columns - 1; i++)
            for (j = 0; j < nodesPerLayer[i]; j++) {
                sum = 0.0;
                for (k = 0; k < nodesPerLayer[i - 1]; k++)
                    sum += neuralNet[i][j].weights[k] * neuralNet[i -
1][k].output;
                neuralNet[i][j].output = 1.0 / (1.0 + Math.exp(-sum));
            }
```

```
        /*output Layer*/
        for (i = 0; i < nodesPerLayer[columns - 1]; i++) {
            sum = 0.0;
            for (j = 0; j < nodesPerLayer[columns - 2]; j++)
                sum += neuralNet[columns - 1][i].weights[j] * neuralNet[columns
- 2][j].output;
            output[i] = neuralNet[columns - 1][i].output = 1.0 / (1.0 +
Math.exp(-sum));
        }
        return output;
    }

    public double computeError(boolean training, double[] extrema) {
        int i, j;
        double[] input = new double[nodesPerLayer[0]];
        double[] output;
        double[] desiredOutput = new double[nodesPerLayer[columns - 1]];
        double totalError = 0.0;
        result = new double[dataSet.length];
        int k = 0;
        for (i = 0; i < dataSet.length; i++) {
            for (j = 0; j < numIns; j++) input[j] = dataSet[i][j];
            for (j = 0; j < numOuts; j++) desiredOutput[j] = dataSet[i][numIns
+ j];
            output = computeOutput(input);

            if (!training) {
                if(k%12 == 0)
                    System.out.println();
                double range = extrema[1] - extrema[0];
                double scaleUp = ((output[0] - .05) / .9) * range;
                System.out.println(extrema[0] + scaleUp);
                result[i] = extrema[0] + scaleUp;
            }
            for (j = 0; j < numOuts; j++) {
                double error = output[j] - desiredOutput[j];
                totalError += error * error;
            }
            k++;
        }
```

```
        return totalError;
    }

    public double[] getResult() {
        return result;
    }
}
```

Class Neuron pada *controller*

```
package app.calculation;

public class Neuron {
    public double output;
    public double[] weights;
    public int numWeights;

    Neuron(int numWeights) {
        this.numWeights = numWeights;
        this.weights = new double[this.numWeights];
    }

    public void calculateOutput(double[] input) {
        output = 0;
        for (int i = 0; i < numWeights; i++) {
            output += weights[i] * input[i];
        }
    }
}
```