SISTEM PREDIKSI KETIDAKDISIPLINAN SISWA MENGGUNAKAN METODE *ARTIFICIAL NEURAL NETWORK* (STUDI KASUS : SMK NEGERI 1 PACITAN)

SKRIPSI

Oleh

Yusuf Eka Sayogana

NIM 132410101052

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2020

SISTEM PREDIKSI KETIDAKDISIPLINAN SISWA MENGGUNAKAN METODE *ARTIFICIAL NEURAL NETWORK* (STUDI KASUS : SMK NEGERI 1 PACITAN)

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syaratuntuk menyelesaikan Program Studi Sistem Informasi (S1)

Oleh

Yusuf Eka Sayogana

NIM 132410101052

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2020

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Allah SWT yang senantiasa memberikan rahmat dan hidayah-Nya untuk mempermudah dan melancarkan dalam mengerjakan skrispsi;
2. Ayahanda Jumeno dan Ibunda Siti Amini;
3. Adik Rakhaby Dwi Cholif Olifian;
4. Fatimatuz Zah'ro atas dukungan beserta doanya;
5. Sahabat-sahabatku dengan dukungan dan doanya;
6. Guru-guruku baik dari Pendidikan formal maupun informal;
7. Almamater Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Jember

MOTO

“Hidup memang tidak adil, jadi biasakan dirimu ya”

-Patrick Star-

“Uang adalah sumber utama kegembiraan”

-Tuan Krab-

“If you are good at something, never do it for free”

-Joker-

“If you only do what you can do, you will never be more than you are now.”

-Master Shifu-

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Yusuf Eka Sayogana

NIM : 132410101052

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "SISTEM PREDIKSI KETIDAKDISIPLINAN SISWA MENGGUNAKAN METODE ARTIFICIAL NEURAL NETWORK (STUDI KASUS : SMK NEGERI 1 PACITAN)", 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 manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, 7 April 2020

Yang menyatakan,

Yusuf Eka Sayogana

NIM 132410101052

PENGESAHAN PEMBIMBING

Skripsi berjudul "Sistem Prediksi Ketidakdisiplinan Siswa Menggunakan Metode Artificial Neural Network (Studi Kasus : SMK Negeri 1 Pacitan)" telah diuji dan disahkan pada :

hari, tanggal : Jumat, 26 Juni 2020

tempat : Fakultas Ilmu Komputer Universitas Jember

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Prof. Dr. Saiful Bukhori, ST., M.Kom	Nelly Oktavia A, S.Si., MT
NIP. 196811131994121001	NIP. 198410242009122008

PENGESAHAN PENGUJI

Skripsi berjudul ”Sistem Prediksi Ketidakdisiplinan Siswa Menggunakan Metode Artificial Neural Network (Studi Kasus : SMK Negeri 1 Pacitan)” karya Yusuf Eka Sayogana telah diuji dan disahkan pada:

hari, tanggal : Jumat, 26 Juni 2020

tempat : Fakultas Ilmu Komputer Universitas Jember.

Disetujui oleh:

Penguji I,	Penguji II,
Achmad Maududie, ST, M.Sc.	Gama Wisnu Fajarianto, S.Kom., M.Kom.
NIP. 197004221995121001	NIP. 760015717

Mengesahkan
Dekan Fakultas Ilmu Komputer

Prof.Dr. Saiful Bukhori, S.T., M.Kom
NIP. 196811131994121001

RINGKASAN

Sistem Prediksi Ketidakdisiplinan Siswa Menggunakan Metode Artificial Neural Network (Studi Kasus : SMK Negeri 1 Pacitan); Yusuf Eka Sayogana, 132410101052; 2020, 80 halaman; Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Jember.

SMKN 1 Pacitan menerapkan suatu sistem informasi pemberian poin pelanggaran terhadap para siswanya yang melakukan pelanggaran tata tertib sekolah untuk memantau tingkat kedisiplinan siswanya. Setiap pelanggaran memiliki bobot tersendiri. Total pelanggaran akan dikalikan dengan bobot dan diakumulasi. Hasil dari akumulasi tersebut menjadi dasar untuk pemberian hukuman terhadap siswa yang memiliki tingkat ketidakdisiplinan tinggi.

Data yang diperoleh dari basis data menunjukkan penurunan jumlah pelanggaran di tahun ketiga setelah sistem diterapkan. Hal ini menunjukkan bahwa penerapan pemberian poin pelanggaran mampu mempengaruhi tingkat kedisiplinan siswa. Namun hal ini tidak dapat mencegah siswa agar tidak melakukan pelanggaran dari awal masuk sekolah. Guru Bimbingan Konseling (BK) selaku pihak yang memiliki tanggung jawab tidak memiliki acuan untuk melakukan prioritas bimbingan. Sehingga perlu adanya sistem prediksi terhadap siswa yang berpotensi melakukan pelanggaran. Dengan adanya prediksi guru BK dapat memiliki prioritas, serta penerimaan siswa baru pun dapat lebih selektif.

Penulis membuat sistem prediksi tingkat ketidakdisiplinan siswa dengan memanfaatkan data-data pelanggaran di tahun sebelumnya menggunakan *Artificial Neural Network*. Data yang diambil adalah data siswa aktif di semester terkahir. Karena data yang didapat memiliki ketidakseimbangan kelas maka dilakukan resampling. Hasil prediksi yang dibuat mendapatkan nilai akurasi sebesar 71.84%, nilai presisi sebesar 65.4%, nilai *recall* sebesar 93.11%, dan nilai *F-Measure* sebesar 76.70%.

PRAKATA

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Sistem Prediksi Ketidakdisiplinan Siswa Menggunakan Metode Artificial Neural Network (Studi Kasus : SMK Negeri 1 Pacitan)". Skripsi ini disusun untuk memenuhi salah satu sarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari dukungan berbagai belah pihak. Oleh karena itu penulis menyampaikan terima kasih kepada:

1. Allah SWT yang senantiasa memberikan rahmat dan hidayah-Nya untuk mempermudah dan melancarkan proses pengerjaan skripsi;
2. Prof. Dr.Saiful Bukhori, ST., M.Kom. selaku dosen pembimbing utama dan dekan Fakultas Ilmu Komputer;
3. Nelly Oktavia Adiwijaya S.Si.,MT. selaku dosen pembimbing pendamping yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Fakultas Ilmu Komputer Universitas Jember;
5. Bapak Jumeno dan Ibu Siti Amini yang selalu mendukung dan mendoakan;
6. Adik Rakhaby Dwi Cholif Olifian;
7. Fatimatuz Zah'ro yang selalu memberi semangat, dukungan serta doa;
8. SMK Negeri 1 Pacitan yang telah bersedia menjadi objek penelitian;
9. Keluarga INTENTION atas kekeluargaan yang selalu hangat dan menenangkan;
10. Teman – teman Program Studi Sistem Informasi di semua angkatan;
11. Guru – guru Pendidikan formal maupun non formal;

12. Semua pihak yang tidak dapat disebutkan satu persatu;

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, 8 April 2020

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, tujuan dan manfaat, batasan masalah, metodologi penelitian, dan sistematika penulisan.

1.1 Latar Belakang

Disiplin adalah suatu sikap atau perilaku yang sesuai dengan aturan atau tata tertib yang berlaku baik itu dengan kemauan diri sendiri ataupun karena takut akan hukuman yang berlaku (Arisana & Ismani, 2012). Kedisiplinan siswa sangat berpengaruh terhadap prestasi siswa. Menurut Arisana (2012) kedisiplinan dari seorang siswa akan berpengaruh positif dan signifikan terhadap prestasi siswa tersebut. Siswa yang memiliki tingkat kedisiplinan tinggi memiliki prestasi yang lebih bagus daripada siswa yang memiliki tingkat kedisiplinan rendah.

SMK Negeri 1 Pacitan telah menerapkan suatu sistem informasi pemberian poin pelanggaran terhadap para siswanya yang melakukan pelanggaran tata tertib sekolah untuk memantau tingkat kedisiplinan siswanya. Setiap pelanggaran memiliki bobot tersendiri. Total pelanggaran akan dikalikan dengan bobot dan diakumulasi. Hasil dari akumulasi tersebut menjadi dasar untuk pemberian hukuman terhadap siswa yang memiliki tingkat ketidakdisiplinan tinggi.

Data yang diperoleh dari basis data menunjukkan penurunan jumlah pelanggaran di tahun ketiga setelah sistem diterapkan. Hal ini menunjukkan bahwa penerapan pemberian poin pelanggaran mampu mempengaruhi tingkat kedisiplinan siswa. Namun hal ini tidak dapat mencegah siswa agar tidak melakukan pelanggaran dari awal masuk sekolah. Guru Bimbingan Konseling (BK) selaku pihak yang memiliki tanggung jawab tidak memiliki acuan untuk melakukan prioritas bimbingan. Sehingga perlu adanya sistem prediksi terhadap siswa yang berpotensi melakukan pelanggaran. Dengan adanya prediksi guru BK dapat memiliki prioritas, serta penerimaan siswa baru pun dapat lebih selektif.

Pada penelitian sebelumnya yang dilakukan oleh Astuti (2015) dengan judul penelitian analisis prediksi tingkat ketidakdisiplinan siswa menggunakan algoritma

naïve bayes classifier (studi kasus : smk negeri 1 pacitan) melakukan pengujian data siswa berjumlah 895 dan menghasilkan akurasi sebesar 79,01%. Berdasarkan penelitian yang dilakukan oleh Sharma (2014) yang membandingkan antara *naive bayes* dan *neuralnetwork* pada klasifikasi untuk deteksi spam pada email menunjukkan bahwa *neural network* menghasilkan nilai akurasi, presisi, dan *recall* yang lebih besar dibandingkan *naive bayes*.

Berdasarkan uraian tersebut penulis membuat sistem prediksi tingkat ketidakdisiplinan siswa dengan memanfaatkan data-data pelanggaran di tahun sebelumnya menggunakan *Artificial Neural Network*

1.2 Rumusan Masalah

Berdasarkan uraian yang telah disampaikan dalam latar belakang mendefinisikan permasalahan yang harus diselesaikan dalam penulisan ini, yaitu :

1. Bagaimana menerapkan metode *Artificial Neural Network* dalam mengklasifikasikan tingkat ketidakdisiplinan siswa SMK Negeri 1 Pacitan?
2. Bagaimana mengetahui performansi klasifikasi dengan menerapkan metode *Artificial Neural Network*?

1.3 Tujuan

Tujuan dalam penelitian ini merupakan jawaban dari rumusan masalah yang telah disampaikan. Adapun tujuan penelitian ini adalah:

1. Mengetahui cara penerapan metode *Artificial Neural Network* dalam mengklasifikasikan tingkat ketidakdisiplinan siswa.
2. Mengetahui uji performansi klasifikasi dalam penerapan *Artificial Neural Network*.

1.4 Batasan Masalah

Penulis memberikan batasan masalah untuk objek dan tema yang dibahas sehingga tidak terjadi penyimpangan dalam proses penulisan dan pembuatan aplikasi.

Berikut adalah batasan masalah yang dicantumkan :

1. Studi kasus yang digunakan adalah SMKN 1 Pacitan.
2. Hanya melakukan prediksi ketidakdisiplinan siswa.
3. Sistem prediksi menggunakan metode *Artificial Neural Network*

BAB 2. TINJAUAN PUSTAKA

Pada bagian ini dipaparkan tinjauan yang berkaitan dengan masalah yang dibahas, kajian teori yang berkaitan dengan masalah serta kajian teori yang dikaitkan dengan permasalahan yang dihadapi.

2.1 Data Mining

Data mining adalah kegiatan mengekstrak informasi atau pengetahuan (*knowledge*) penting dari suatu set data berukuran besar dengan menggunakan teknik tertentu. Dinamakan data mining karena proses penemuan informasi atau knowledge dalam set data dilakukan seperti orang melakukan penambangan (Santoso & Umam, 2018).

(Han, Kamber, & Pei, 2012) mengatakan bahwa data mining adalah proses menemukan pola dan pengetahuan dari sejumlah data. Sumber data dapat mencakup database, data warehouses, web, repositori informasi lainnya.

Dalam *data mining* terdapat beberapa tahap yaitu :

1. *Data cleaning*
2. *Data integration*
3. *Data selection*
4. *Data transformation*
5. *Data mining*
6. *Pattern evaluation*
7. *Knowledge presentation*

(Santoso & Umam, 2018) mengatakan tugas-tugas yang biasa dilakukan oleh *data mining* antara lain:

1. *Clustering*

   Mengelompokkan objek ke dalam beberapa kelompok berdasarkan kemiripan antar objek. *Clustering* tidak memerlukan data pelatihan yang sudah diberi label.

2. Klasifikasi

   Melakukan pengelompokan objek berdasarkan kelompok yang sudah ada. Klasifikasi memerlukan data latihan yang sudah diberi label.

3. Regresi

Regresi pada dasarnya mirip dengan klasifikasi, yaitu memerlukan data latihan yang sudah diberi label. Dalam klasifikasi label berupa nilai diskrit sedangkan dalam regresi berupa nilai kontinyu.

4. Asosiasi

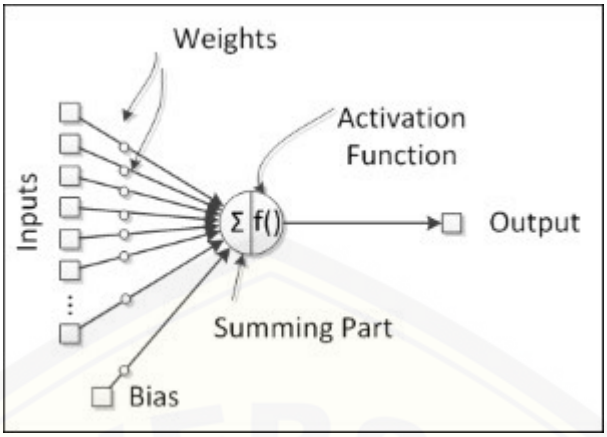
Melakukan asosiasi antar objek dalam suatu set data. Asosiasi dilakukan dengan menghitung berapa kali dalam set data suatu transaksi yang mengandung lebih dari 2 item yang berhubungan. Sering disebut dengan *Market Basket Analysis.*

Dalam penelitian ini penulis melakukan klasifikasi siswa dengan menggunakan label disiplin dan tidak disiplin.

2.2 Artificial Neural Network

Artificial Neural Network (ANN) merupakan algoritma yang terinspirasi dari sistem jaringan syaraf makhluk hidup. ANN meniru cara kerja otak manusia dengan karakteristik mengingat, menghitung, menggeneralisasi, mengadaptasi dengan konsumsi energi yang rendah. ANN mempunyai beberapa kelebihan yaitu kemampuan menyelesaikan prediksi yang non linier, waktu penyelesaian yang cepat, *robust* terhadap *missing data* (Santoso & Umam, 2018).

ANN adalah sebuah struktur untuk menjalankan berbagai macam tugas seperti *pattern recognition* atau pengenalan pola, pengkajian terhadap data dan peramalan tren yang merupakan tugas-tugas yang dapat dilakukan seorang ahli dengan keilmuan yang dimiliki. ANN merupakan kebalikan dari pendekatan algoritma konvensional yang membutuhkan serangkaian langkah yang harus dilakukan untuk mencapai tujuan tertentu. ANN memiliki kemampuan untuk mempelajari cara untuk menyelesaikan suatu tugas dengan sendirinya karena ANN memiliki struktur jaringan yang memiliki interkoneksi yang tinggi (Soares & Souza, 2016).



Gambar 2.1 Elemen dasar *artificial neural network* (Soares & Souza, 2016)

Gambar 2.1 merupakan gambaran elemen dasar dari *artificial neural network* (ANN). Elemen tersebut adalah input, bobot, bias, *summing part*, *activation function*, dan output.

2.2.1 Fungsi Aktivasi

Hasil dari neuron yang bekerja pada ANN berasal dari sebuah fungsi aktivasi. Komponen ini memberikan kondisi non linier pada proses neural network yang dibutuhkan karena neuron secara alami memiliki sifat nonlinier. Fungsi aktivasi dibatasi menjadi dua nilai pada luarannya, oleh karena itu menjadi fungsi non linear, tetapi dalam beberapa kasus bisa menjadi fungsi linear (Soares & Souza, 2016).

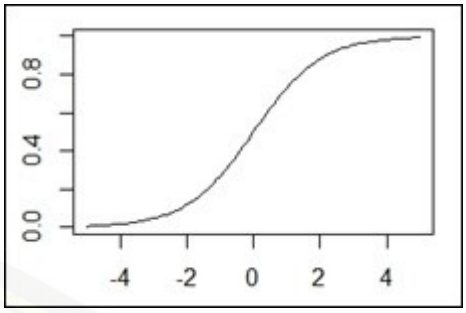
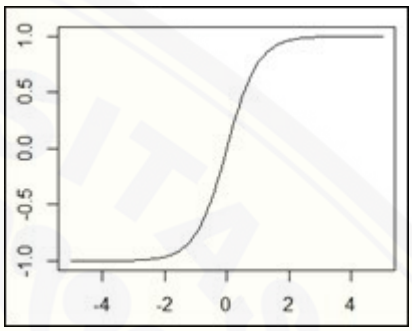
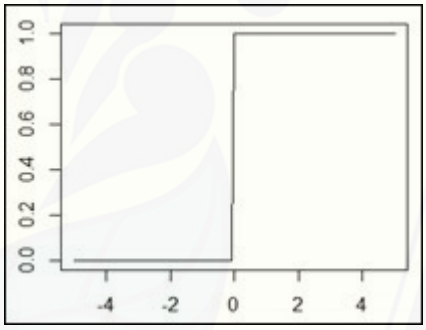
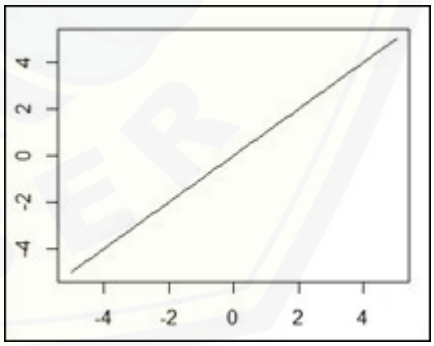
Empat fungsi aktivasi yang paling sering digunakan sebagai berikut :

1. Sigmoid
2. Hyperbolic tangent
3. Hard limiting threshold
4. Purely linear (Setiawan, Putri, & Suryanita, 2019)

Berikut merupakan table rumus dan gambaran dari masing-masing fungsi aktivasi:

Tabel 2.1. Daftar fungsi aktivasi (Soares & Souza, 2016)

Function	Equation	Chart

Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
Hyperbolic tangent	$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$	
Hard limiting threshold	$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x \geq 1 \end{cases}$	
Purely linear	$f(x) = x$	

2.2.2 Weights

Dalam Neural Network, bobot merepresentasikan hubungan antar neuron dan memiliki kemampuan untuk memperkuat atau melemahkan sinyal neuron seperti menggandakan sinyal, hingga mengubah sinyal. Bobot dari saraf atau neural memiliki kemampuan untuk mempengaruhi luaran dari neuron. Oleh karena itu

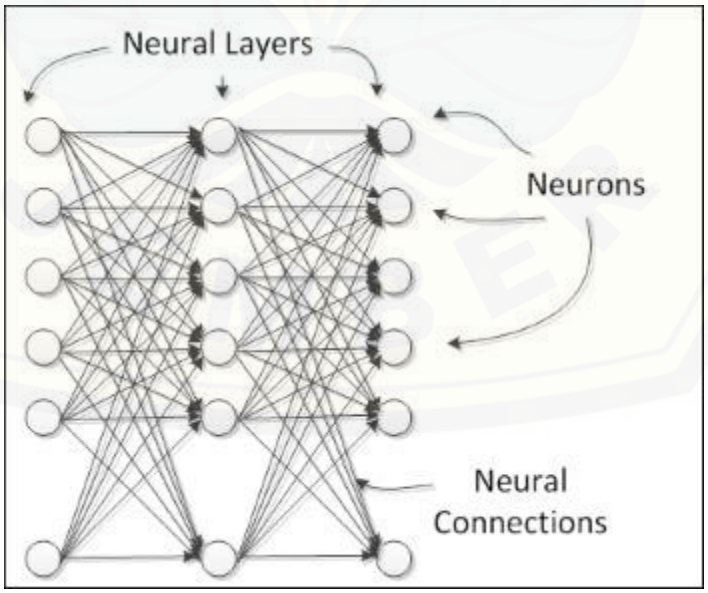
proses aktivasi neuron sangat bergantung pada masukan dan bobot (Soares & Souza, 2016).

2.2.3 Bias

Neuron bisa memiliki komponen independen yang dapat menambahkan sinyal pada fungsi aktivasi. Komponen ini dinamakan bias. Seperti halnya input, bias juga memiliki bobot. Fitur ini dapat membantu pengetahuan neural network sebagai sistem non linier yang lebih murni (Soares & Souza, 2016).

2.2.4 Layers

Jaringan syaraf alami terorganisir dalam bentuk lapisan, setiap lapisan menyediakan tingkatan proses tertentu, seperti lapisan *input* menerima rangsangan langsung dari dunia luar dan lapisan *output* memicu aksi yang akan memberikan efek langsung pada dunia luar. Diantara lapisan-lapisan tersebut, terdapat beberapa lapisan tersembunyi yang berarti lapisan tersebut tidak berhubungan langsung dengan dunia luar. Pada *artificial neural networks*, semua neuron pada sebuah lapisan memiliki *input* dan fungsi aktivasi yang sama seperti ditunjukkan pada gambar berikut :



Gambar 2.2 Layer dalam neural network (Soares & Souza, 2016)

Gambar 2.2 menjelaskan tentang gambaran layer dalam neural network. Layer berisikan beberapa neuron. Tiap neuron akan terhubung ke neuron lain di layer selanjutnya.

Nural networks dapat tersusun dari beberapa lapisan yang saling terhubung membentuk jaringan yang disebut *multilayer*. Lapisan saraf pada dasarnya dapat dibagi menjadi tiga kelas:

1. Input layer
2. Hidden layer
3. Output layer

Dalam praktiknya, lapisan saraf tambahan menambah tingkat abstraksi rangsangan luar, sehingga meningkatkan kapasitas jaringan saraf untuk mewakili pengetahuan yang lebih kompleks.

2.2.5 Arsitektur Neural Network

Pada dasarnya, jaringan saraf dapat memiliki tata letak yang berbeda tergantung pada bagaimana neuron atau lapisan neuron saling terhubung. Setiap arsitektur jaringan saraf dirancang untuk tujuan tertentu. Jaringan saraf dapat diterapkan pada sejumlah masalah dan tergantung pada sifat masalah, jaringan saraf harus dirancang untuk mengatasi masalah dengan lebih efisien. (Soares & Souza, 2016)

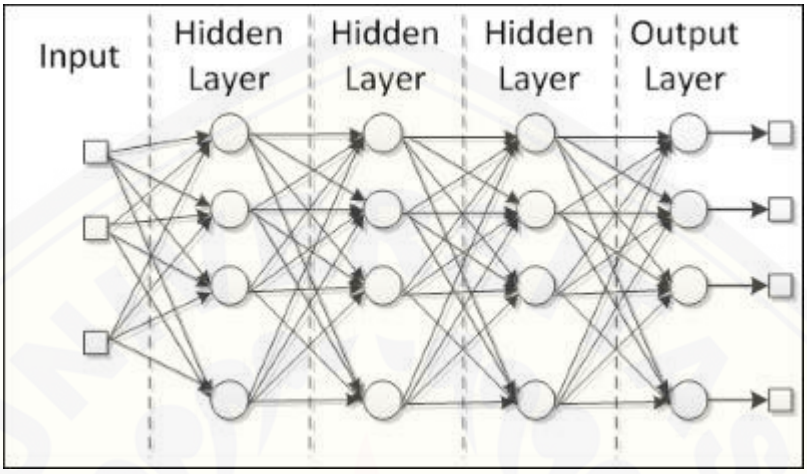
Ada dua modalitas arsitektur untuk jaringan saraf:

1. Neuron connections
    - Monolayer networks
    - Multilayer networks
2. Signal flow
    - Feedforward networks
    - Feedback networks

Di dalam penelitian ini penulis menggunakan arsitektur *multilayer* dan *feedforward*

2.2.6 Multilayer networks

Dalam kategori ini, neuron dibagi menjadi beberapa lapisan, seperti yang ditunjukkan pada gambar berikut:



Gambar 2.3 Multilayer networks (Soares & Souza, 2016)

Gambar 2.3 merupakan gambaran sebuah multilayer neural network. Multilayer memiliki minimal 1 hidden layer didalamnya.

2.2.7 Feedforward networks

Aliran sinyal dalam jaringan saraf dapat berupa sinyal satu arah atau dalam pengulangan. Dalam kasus pertama, kami menyebutnya arsitektur jaringan *neural feedforward*, karena sinyal input dimasukkan ke dalam lapisan input; kemudian, setelah diproses, mereka diteruskan ke lapisan berikutnya, seperti yang ditunjukkan pada gambar di bagian *multilayer* (Soares & Souza, 2016).

2.2.8 Back Propagation

Salah satu metode untuk melakukan *training multilayer neural network* adalah algoritma *back propagation*. Algoritma *back propagation* dapat optimasi menggunakan *learning rule gradient descent*. Algoritma ini sangat bermanfaat, cukup andal, dan mudah dipahami. Selain itu banyak algoritma lain yang mendasarkan prosesnya pada *back propagation* (Santoso & Umam, 2018).

Algoritma *backpropagation* melakukan pelatihan *multilayer perceptron* dalam 2 tahap, yaitu : perhitungan maju untuk menghitung *lost function* antara

keluaran aktual dan target; dan perhitungan mundur yang mempropagasikan balik galat tersebut untuk memperbaiki bobot-bobot sinaptik pada semua neuron yang ada (Suyanto, 2018).

Menurut (Suyanto, 2018) alur *back propagation* adalah sebagai berikut:

A. Inisialisasi arsitektur jaringan.

Penentuan nilai learning rate, kondisi berhenti, dan bobot-bobot sinaptik secara acak

B. *Repeat*

- Perhitungan maju
- Perhitungan loss function
- Perhitungan mundur
- Update bobot

Until kondisi berhasil dipenuhi

2.2.9 Gradient Descent

Metode ini menjadi dasar algoritma back propagation yang sering digunakan untuk melakukan *trainingneuralnetwork* dengan banyak neuron (Santoso & Umam, 2018).

Rumus perhitungan jumlah error :

$$E(w) = \frac{1}{2}\sum_{t \in T}(d_t - y_t)^2$$

Rumus turunan E terhadap bobot :

$$\nabla E(w) = [\ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots \frac{\partial E}{\partial m}]$$

Rumus update bobot:

$$w \leftarrow w + \Delta w$$

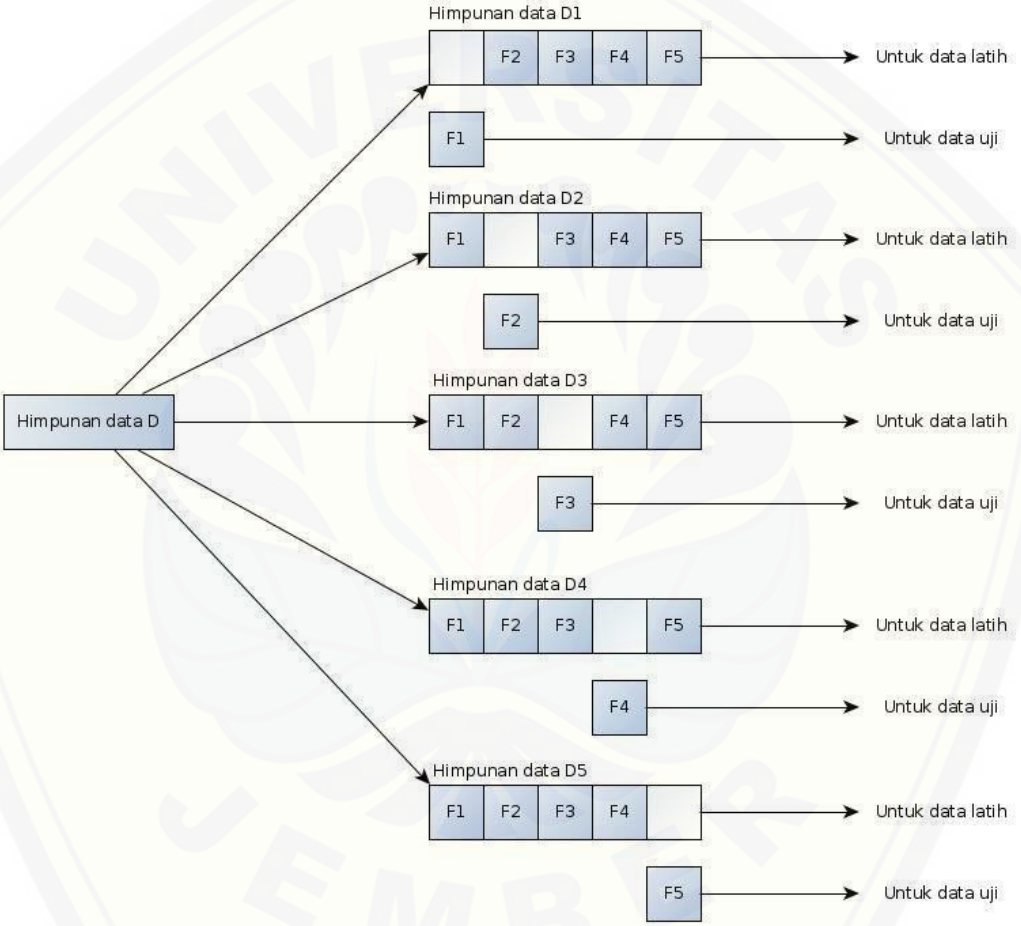
$$\Delta w = -\eta \nabla E(w)$$

2.3 K-Fold Cross Validation

Cross validation adalah metode yang digunakan untuk mengestimasi kesalahan prediksi untuk evaluasi kinerja model. Dalam k-fold cross validation,

data dibagi menjadi k bagian dengan ukuran yang hampir sama. Klasifikasi model di *training* dan testing sebanyak k kali. Setiap pengulangan, satu bagian akan digunakan sebagai data *testing* dan sisanya digunakan untuk *training* (Nurhayati, Soekarno, & Hadihardaja, 2014).

Pada umumnya metode *k fold cross validation* menggunakan k=10 untuk mendapatkan akurasi dengan bias dan variansi yang relatif rendah (Suyanto, 2018).



Gambar 2.4 *K-Fold Cross Validation* (Suyanto, 2018).

Gambar 2.4 adalah gambaran dari pengelompokan data dalam proses k fold cross validation. Dalam setiap bagian akan dijadikan data uji secara bergiliran. Bagian yang tidak menjadi data uji akan menjadi data latih.

BAB 3. METODOLOGI PENELITIAN

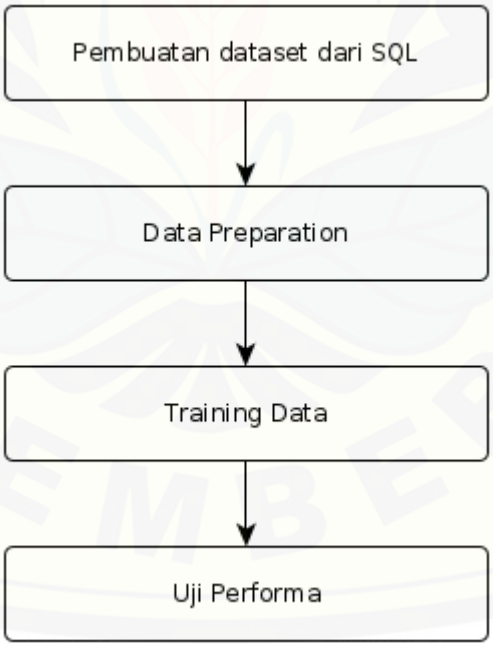
Bab ini menggambarkan tentang penelitian yang akan dilakukan untuk menjawab rumusan masalah sehingga dapat mewujudkan tujuan sebenarnya dari penelitian. Pada metodologi penelitian akan dijelaskan tentang jenis penelitian, tempat dan waktu penelitian serta tahapan dari penelitian.

3.1 Metode Penelitian

Metode penelitian yang dilakukan merupakan penelitian pengembangan. Penelitian pengembangan bertujuan untuk mengembangkan produk tertentu dan menguji keefektifan produk tersebut. Penelitian ini mengembangkan prediksi ketidakdisiplinan siswa menggunakan *artificiaheural network*

3.2 Tahapan Penelitian

Berikut merupakan tahapan penelitian dari prediksi ketidakdisiplinan siswa:



Gambar 3.1 Alur tahapan

Gambar 3.1 menggambarkan alur dari penelitian yang dilakukan, Berikut penjelesan dari masing-masing alur:

Tabel 3.1. Tahapan Klasifikasi

No	Tahapan	Input	Proses	Output
1	Pembuatan Dataset	SQL Database	Pemilihan atribut data. Melakukan *Query* pengambilan data siswa. Pemberian kelas label.	Dataset
2	Persiapan Data	Dataset	Penanganan missing value. Penanganan data redundant. Mengubah data string ke numeric. Melakukan resampling	Data Training , Data Testing
3	Proses Training	Data Training , Data Testing	Implemenasi neural network	Model
4	Uji Performa	Model	k-fold cross validation	Hasil Performa

3.2.1 Pembuatan Dataset

Dalam tahapan pembuatan *dataset* proses pertama yang dilakukan adalah pemilihan atribut *dataset*. Atribut yang dipilih dari basis data SMKN 1 Pacitan untuk menjadi atribut dalam *dataset* adalah jenis kelamin, asal SMP, tanggal lahir, jurusan dan pekerjaan orang tua. Atribut tanggal lahir di proses menjadi atribut lebih tua yang digunakan untuk memeriksa kesesuaian umur siswa dengan tingkatan kelas siswa.

Menurut (Muschkin, 2014) murid yang memiliki usia yang lebih tua dari usia pada suatu tingkatan kelas yang seharusnya memiliki kecenderungan untuk berbuat kenakalan dan dikeluarkan dari sekolah. Hal ini menjadi alasan untuk memasukkan atribut lebih tua ke dalam *dataset*

Atribut jenis kelamin dipilih menjadi atribut *dataset* karena dalam beberapa semester terdapat kecenderungan jenis kelamin laki-laki melakukan pelanggaran

yang lebih banyak dibanding perempuan. Atribut pekerjaan orang tua , asal sekolah, dan jurusan pun dipilih karena terdapat kecenderungan melakukan pelanggaran.

Pemberian *class label* pada *dataset* didapatkan dari hasil akumulasi poin siswa selama satu semester. Siswa dinyatakan tidak disiplin jika jumlah akumulasi poin lebih dari 30. Hal tersebut didasarkan pada surat keputusan kepala Sekolah Menengah Kejuruan Negeri 1 Pacitan nomor 421.5/474/101.6.20.9/2018 yang menyebutkan peringatan pertama akan diberikan saat siswa mendapatkan poin lebih dari 30.

Proses pembuatan dataset dilakukan dengan melakukan *query* ke basis data sistem informasi SMK Negeri 1 Pacitan. *Query* merupakan perintah yang digunakan untuk mengakses basis data. *Query* yang dilakukan untuk pembuatan *dataset* adalah *query select* siswa aktif di semester terakhir.

3.2.2 Preprocessing Data

Data yang didapat dari hasil *query* belum dapat digunakan untuk proses *training*. Terdapat beberapa data yang tidak memiliki nilai (*missing value*), dan beberapa data yang memiliki makna sama. Data yang memiliki makna sama akan digabungkan menjadi satu nama, sedangkan data kosong akan diisi dengan nilai modus.

Terdapat 1 atribut yang berpotensi memiliki data kosong, yaitu atribut *parent's job*. Hal ini disebabkan karena kolom pekerjaan di tabel mt_siswa memiliki *default value null*. Nilai modus didapat dengan melakukan *query grouping*. Seluruh atribut parent_job yang kosong akan diubah menjadi nilai modus.

Terdapat dua atribut yang memiliki potensi data bermakna sama, yaitu atribut pekerjaan dan atribut asal sekolah. Dua atribut tersebut disimpan di basis data dalam bentuk *string* tanpa relasi ke tabel lain. Data yang memiliki makna sama dikelompokkan dalam tabel berikut.

Atribut pekerjaan disesuaikan dengan daftar pekerjaan yang digunakan oleh sistem kependudukan yang didapatkan dari situs DISPENDUKCAPIL Kabupaten Rembang dengan alamat *url* http://dindukcapil.rembangkab.go.id/data/pekerjaan.

Karena neural network hanya menggunakan data numerik dalam perhitungannya, maka data non numerik perlu diubah terlebih dahulu. Terdapat dua atribut yang memiliki nilai string, yaitu parent_job dan junior_high_school.

Atribut pekerjaan orang tua diubah sesuai dengan urutan yang ada dalam DISPENDUK. Sedangkan Atribut asal sekolah diubah menjadi nilai urutan berdasarkan basis data.

Data hasil covert memiliki nominal yang besar sehingga akan menyebabkan proses perhitungan berjalan lama. Untuk memudahkan proses perhitungan, perlu dilaksanakan proses scaling. Proses *scaling* adalah proses mengubah data menjadi angka dalam range tertentu. Penulis melakukan scaling data menjadi range 0 - 1. Rumus untuk melakukan scaling adalah sebagai berikut:

$$\hat{X} = \frac{X - Xmin}{Xmax - Xmin} * (BA - BB) + BB$$

BA : Batas Atas

BB : Batas Bawah

3.2.3 Training Data

Implemntasi neural network dilakukan dengan menggunakan backpropagation. Backpropagation dipilih karena mampu menangani masalah non linier. Berdasarkan penelitian yang dilakukan oleh Setiawan (2019) yang membandingkan algoritma genetika dan backpropagation pada aplikasi prediksi penyakit autoimun, backpropagation memperoleh hasil yang lebih baik.

3.2.4 Uji Performa

Uji performa dilakukan dengan melakukan pengujian terhadap nilai hasil prediksi dengan nilai sesungguhnya yang ada di dataset. Uji performansi mencocokkan antara kedua nilai tersebut. Nilai yang akan digunakan untuk mengukur performa dari model yang dibuat adalah nilai akurasi, presisi, *recall* dan nilai F- Measure.

Nilai akurasi merupakan nilai rasio yang membandingkan nilai prediksi yang benar terhadap keseluruhan data yang diprediksi. Berikut adalah rumus untuk mendapatkan nilai akurasi:

$$akurasi = \frac{true\ positive\quad (TP) + \ true\ negative (TN)}{TP + false\ positive + TN + false\ negative}$$

Nilai presisi merupakan nilai rasio yang membandingkan nilai prediksi positif yang benar terhadap keseluruhan data yang diprediksi sebagai positif. Berikut adalah rumus untuk mendapatkan nilai presisi:

$$akurasi = \frac{true\ positive\quad (TP)}{TP + false\ positive}$$

Nilai recall merupakan nilai rasio yang membandingkan nilai prediksi positif yang benar terhadap keseluruhan data positif. Berikut adalah rumus untuk mendapatkan nilai recall:

$$recall = \frac{true\ positive\quad (TP)}{TP + false\ negative}$$

Nilai *F-Measure* merupakan nilai perbandingan rata-rata presisi dan recall yang dibobotkan. Menurut (Saifudin & Wahono, 2015) *F-Measure* adalah metrik evaluasi yang populer untuk masalah ketidakseimbangan. *F-Measure* mengkombinasikan recall dan presisi sehingga menghasilkan metrik yang efektif untuk pencarian kembali informasi dalam himpunan yang mengandung masalah ketidakseimbangan. Berikut adalah rumus untuk mendapatkan nilai *F-Measure*:

$$F - Measure = \frac{2 * recall * presisi}{recall + presisi}$$

3.3 Sumber Data

Data yang digunakan untuk penelitian ini adalah data yang didapat langsung dari basis data sistem informasi manajemen yang ada di SMK Negeri 1 Pacitan. Data berupa berkas sql yang berisi data diri dan data pelanggaran yang dilakukan.

BAB 4. DESAIN DAN IMPLEMENTASI

Bab ini akan membahas tentang perancangan dan penerapan metode *artificial neural network* pada sistem prediksi ketidakdisiplinan siswa. Proses perancangan sistem dimulai dari analisis kebutuhan fungsional dan non – fungsional sistem, pembuatan desain sistem, implementasi sistem, dan pengujian.

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dalam penelitian ini dilakukan dengan cara mengidentifikasi permasalahan dalam melakukan klasifikasi terhadap prediksi tingkat kedisiplinan siswa menggunakan metode Neural Network, kemudian dilakukan proses pencatatan dan analisis untuk menentukan kebutuhan fungsional dan non fungsional sistem.

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional berisi proses yang akan dilakukan oleh sistem. Kebutuhan fungsional dari sistem ini adalah sebagai berikut :

1. Sistem dapat mengolah database yang sudah ada menjadi sebuah dataset
2. Sistem dapat melakukan *pre proccessing data* dari dataset yang telah terbuat
3. Sistem dapat menampilkan hasil klasifikasi

4.1.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan hal yang dibutuhkan oleh sistem untuk mendukung aktivitas sistem sesuai dengan kebutuhan fungsional yang telah disusun. Kebutuhan non-fungsional sistem ini yaitu :

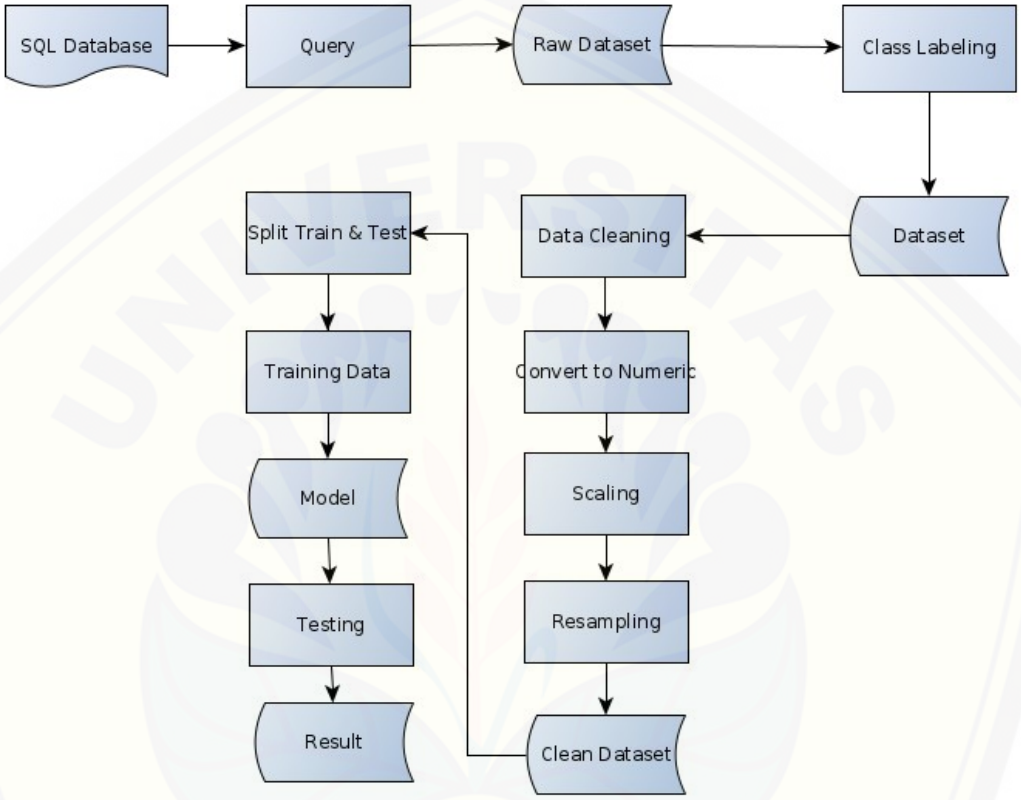
1. Sistem berbasis website yang mudah untuk digunakan agar pengguna tidak kesulitan dalam menggunakan sistem tersebut.
2. Sistem menggunakan autentikasi berupa emai dan password.
3. Tampilan website yang responsive.

4.2 Desain Sistem

Desain sistem yang dibuat meliputi Arsitektur, Flowchart, dan entity relationship diagram (ERD).

4.2.1 Arsitektur

Arsitektur proses klasifikasi ketidakdisiplinan siswa ini menggunakan beberapa komponen yang diintegrasikan, seperti yang diilustrasikan pada gambar di bawah ini



Gambar 4.1 Arsitektur proses klasifikasi

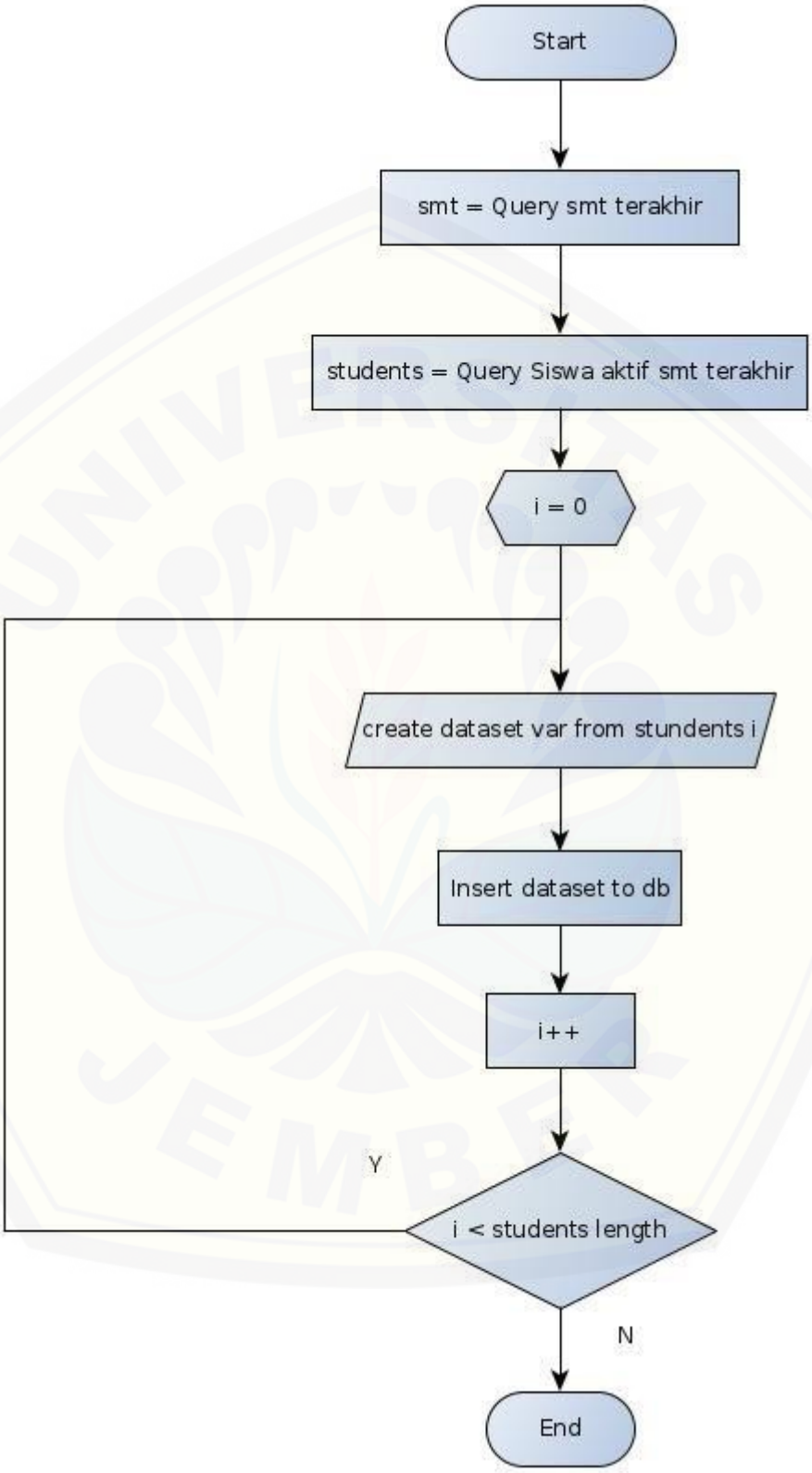
Gambar 4.1 adalah gambaran umum alur dari proses klasifikasi. Proses klasifikasi diawali dengan didapatkannya data berupa sql. Data sql tersebut dilakukan query untuk menghasilkan dataset mentah. Dataset mentah dilakukan *class labeling* untuk menentukan *class* dari setiap dataset. Dataset yang sudah memiliki *class label* kemudian dilakukan preprocessing data yaitu data cleaning, convert data, scaling, dan resampling yang menghasilkan dataset siap pakai. Dataset kemudian dipecah menjadi 10 bagian, 9 bagian digunakan untuk training. Hasil training disimpan dalam model, kemudian dites menggunakan 1 bagian dari dataset untuk menentukan hasil performanya.

4.3 Flowchart

Flowchart merupakan suatu bagan yang digambarkan dengan simbol-simbol tertentu yang bertujuan untuk menggambarkan sebuah alur proses dari sebuah algoritma.

4.3.1 Flowchart Pembuatan Dataset

Flowchart pembuatan *dataset* menggambarkan alur proses pembuatan *dataset* sebagai sumber data pada proses penelitian ini. *Dataset* dibuat merupakan dari basis data sistem informasi SMK Negeri 1 Pacitan.

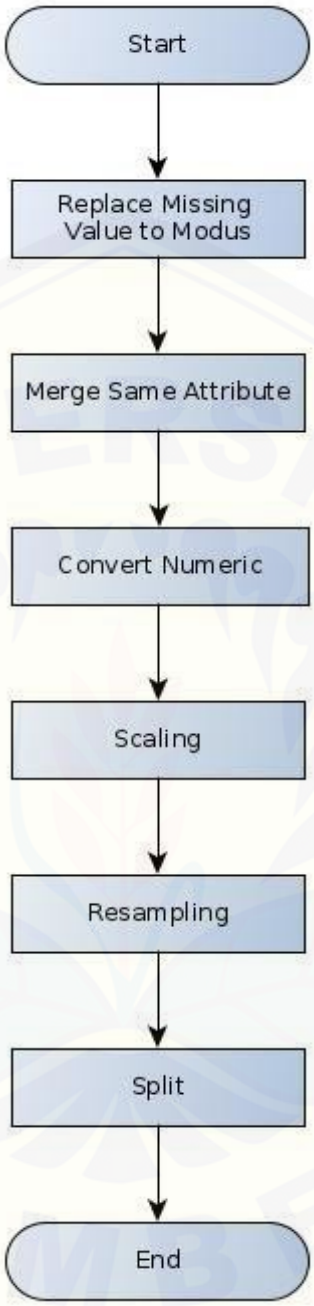


Gambar 4.2 Flowchart Pembuatan Dataset

Gambar 4.2 adalah gambar dari proses pembuatan dataset. Proses dimulai dengan membuat variabel smt dengan *value* mengambil data semester terakhir dari database. Selanjutnya membuat variabel students dengan *value* data siswa aktif di semester terakhir yang didapatkan sebelumnya. Data students kemudian dilakukan perulangan, setiap perulangan diproses menjadi dataset lalu disimpan ke dalam basis data.

4.3.2 Flowchart Preprocessing Data

Flowchart preprocessing data menggambarkan alur proses pengolahan data sebelum proses klasifikasi menggunakan metode Neural Network, Flowchart Preprocessing dapat dilihat pada gambar berikut:



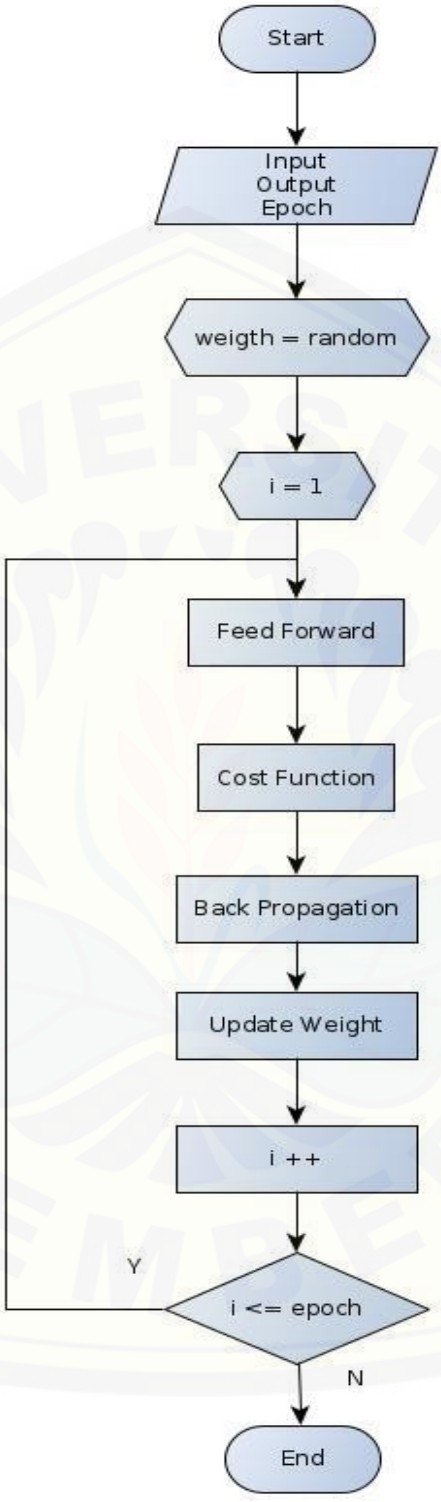
Gambar 4.3 Flowchart Preprocessing Data

Gambar 4.3 adalah gambar *flowchart* dari proses preprocessing data. Proses diawali dengan melakukan penanganan terhadap data kosong dengan menggantinya menjadi nilai modus. Proses kedua adalah melakukan penggabungan attribut yang memiliki makna sama. Proses ketiga adalah mengubah data yang bertipe string menjadi data numerik. Proses keempat adalah mengubah skala data numerik

menjadi skala 0-1. Proses kelima adalah proses resampling untuk menangani ketidakseimbangan data. Proses terakhir adalah proses pembagian data untuk data latih dan data uji.

4.3.3 Flowchart Training Data

Flowchart Proses training menggambarkan proses training mulai dari proses pemberian bobot, perhitungan maju dan mundur, hingga update bobot. Flowchart training dapat dilihat pada gambar berikut:



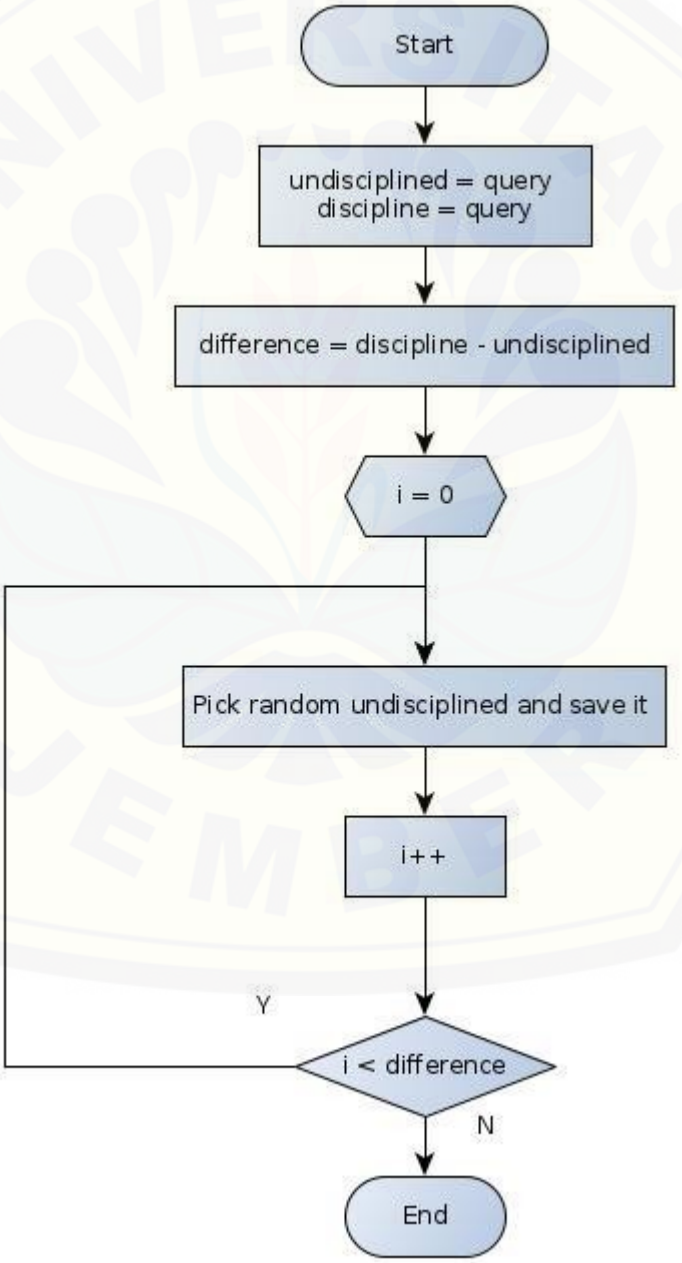
Gambar 4.4 Flowchart Training Data

Gambar 4.4 adalah gambar *flowcart* dari proses training. Proses diawali dengan memberikan nilai input, output, dan epoch, dilanjutkan dengan inisialisasi

nilai bobot secara random. Proses selanjutnya adalah proses perulangan sebanyak epoch yang sudah ditentukan. Di dalam perulangan terdapat proses perhitungan maju, perhitungan jumlah error, perhitungan mundur, dan pembaruan nilai bobot.

4.3.4 Flowchart Random Over Sampling

Flowchart random over sampling adalah flowcart yang menggambarkan proses resampling dengan mengambil data yang tidak seimbang untuk di duplikat.

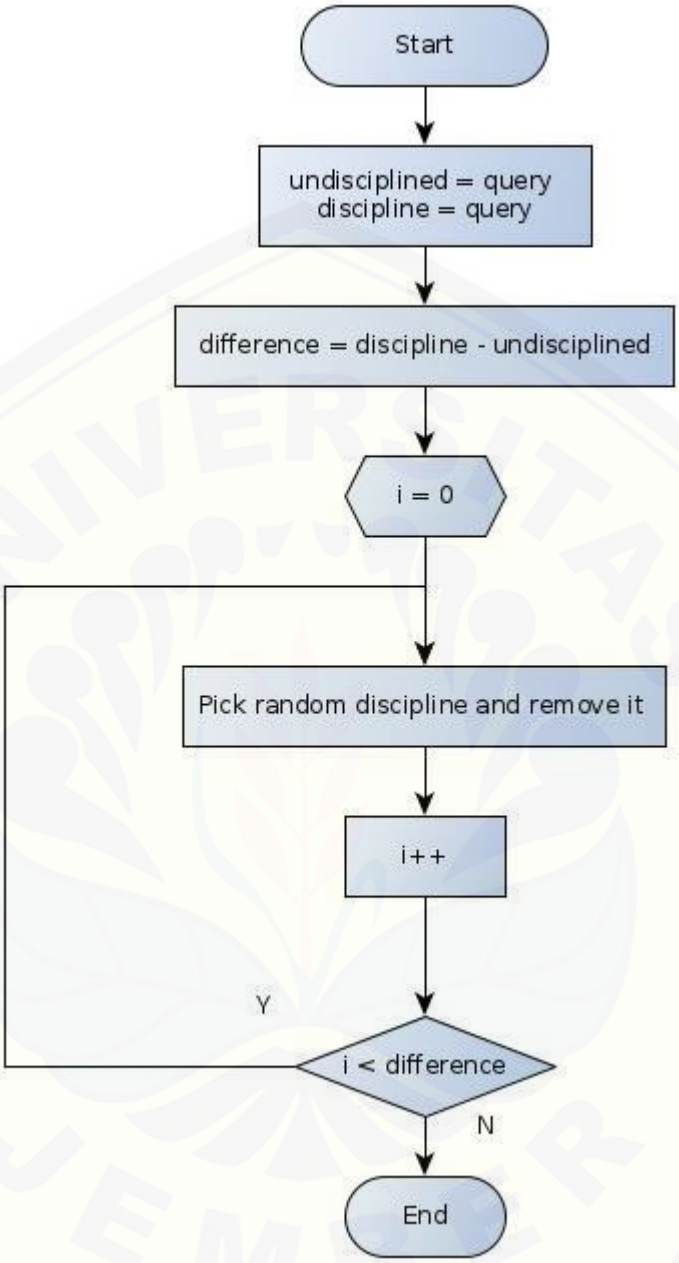


Gambar 4.5 Flowchart Random Over Sampling

Gambar 4.5 adalah gambar *flowcart* dari proses random over sampling. Proses dimulai dengan mengambil data disiplin dan tidak disiplin dari database. Dari kedua data tersebut dilakukan perhitungan nilai selisih. Proses selanjutnya adalah melakukan perulangan sebanyak nilai selisih. Di setiap perulangan dilakukan pengambilan data tidak disiplin secara acak kemudian disimpan dalam database.

4.3.5 Flowchart Random Under Sampling

Flowchart *random over sampling* adalah flowcart yang menggambarkan proses resampling dengan menghapus data yang tidak seimbang.

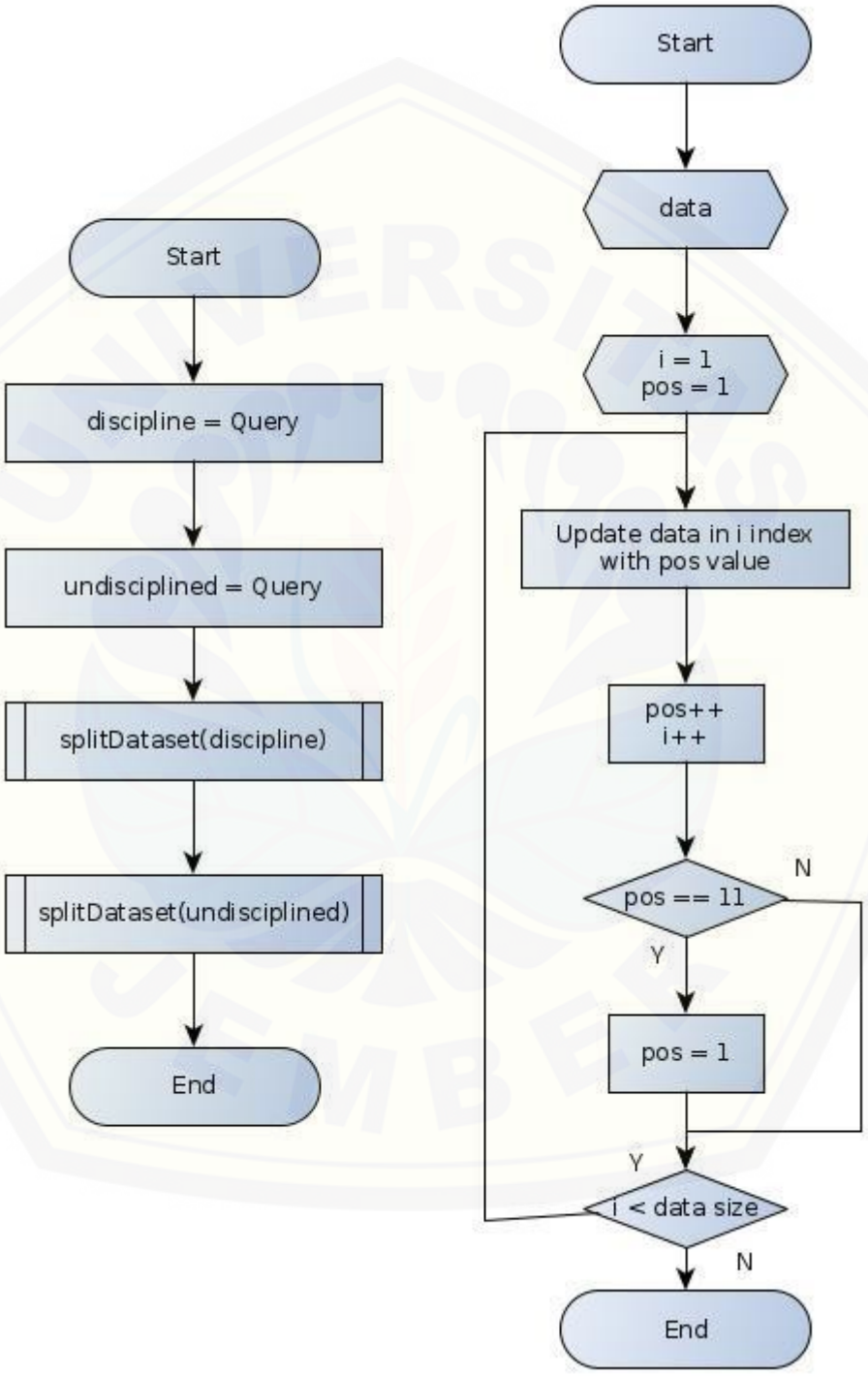


Gambar 4.6 Flowchart Random Under Sampling.

Gambar 4.6 adalah gambar *flowcart* dari proses random under sampling. Proses dimulai dengan mengambil data disiplin dan tidak disiplin dari database. Dari kedua data tersebut dilakukan perhitungan nilai selisih. Proses selanjutnya adalah melakukan perulangan sebanyak nilai selisih. Di setiap perulangan dilakukan penghapusan data disiplin secara acak.

4.3.6 Flowchart Split Data

Flowcart split data adalah flowcart yang menggambarkan proses pembagian data dalam *k-fold cross validation*



Gambar 4.7 Flowcart split dataset

Gambar 4.7 adalah gambar alur dari proses pemisahan data. Terdapat 2 proses dalam gambar 4.7. Proses pertama yaitu proses pengambilan data yang terdapat pada gambar sebelah kiri. Dalam proses tersebut diawali dengan pengambilan data dengan kelas disiplin, selanjutnya pengambilan data dengan kelas tidak disiplin. Kedua data tersebut dilakukan proses *split* dengan memanggil fungsi splitDataset.

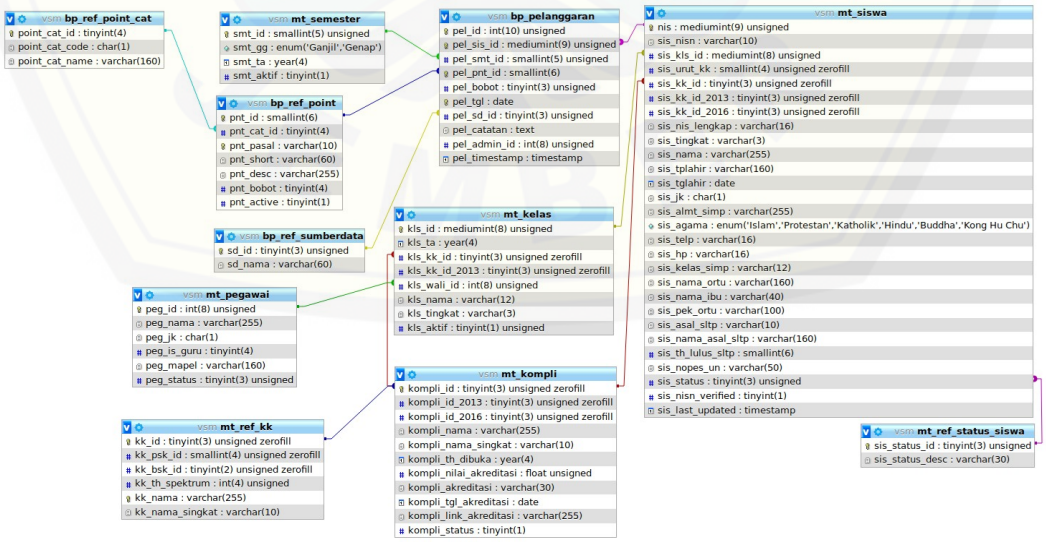
Fungsi proses splitDataset digambarkan dalam gambar sebelah kanan. Proses dimulai dengan menentukan nilai variabel pos, i, dan data dari parameter. Data yang didapat dari parameter dilakukan perulangan. Setiap perulangan dilakukan proses update sesuai dengan nilai variabel pos.

4.4 Entity Relationship Diagram

Entity Relationship Diagram merupakan diagram yang digunakan untuk pemodelan kebutuhan basis data, ERD dari sistem klasifikasi tingkat kedisilpinan siswa pada penelitian ini dapat dilihat pada gambar dibawah ini

4.4.1 ERD Sistem Informasi SMK Negeri 1 Pacitan

ERD Sistem informasi SMK Negeri 1 Pacitan adalah ERD yang didapat dari sistem informasi SMK Negeri 1 pacitan yang digunakan untuk mengelola data pelanggaran siswa.

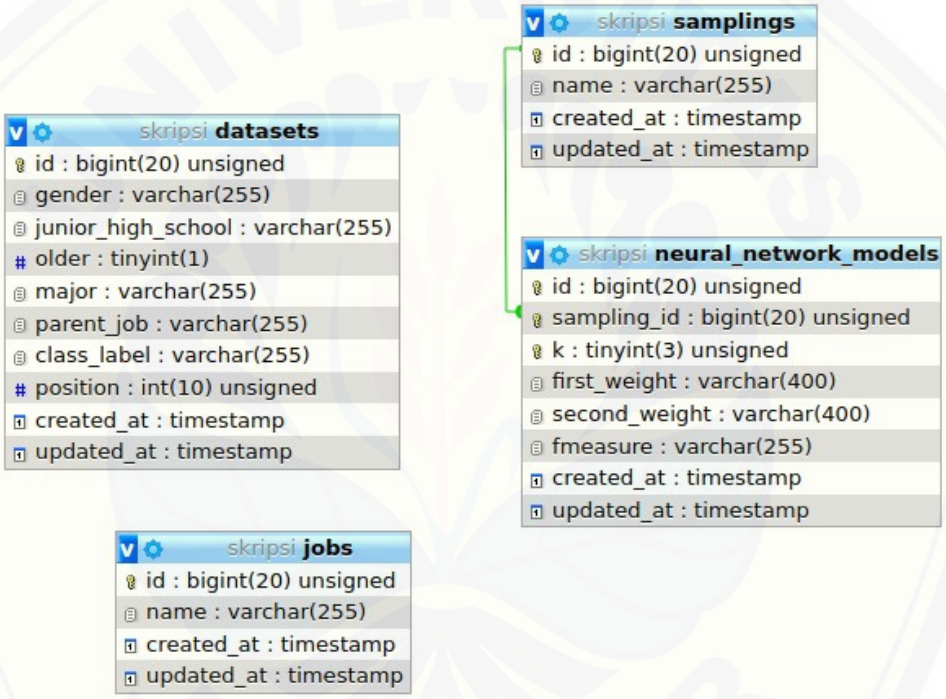


Gambar 4.8 ERD Sistem Informasi SMK Negeri 1 Pacitan

Gambar 4.8 adalah gambar ERD dari sistem informasi yang digunakan di SMK Negeri 1 Pacitan. Dalam ERD ini terdapat detail dari seluruh tabel yang nantinya digunakan sebagai bahan pembuatan dataset.

4.4.2 ERD Sistem prediksi

ERD sistem prediksi adalah ERD yang digunakan untuk mengelola dataset dan *neural network model* yang digunakan untuk keperluan prediksi ketidakdisiplinan siswa.



Gambar 4.9 ERD Sistem Prediksi

Gambar 4.9 adalah ERD dari sistem prediksi. Terdapat 4 tabel di dalam ERD terserbut. Tabel datasets digunakan untuk menyimpan dataset yang dihasilkan dari database sistem informasi SMK Negeri 1 Pacitan. Tabel sampling digunakan untuk menyimpang data jenis sampling. Tabel neural_network_models digunakan untuk menyimpan bobot dari proses training.

4.5 Implementasi Sistem

Sistem yang dikembangkan dalam penelitian ini menggunakan bahasa pemrograman PHP dengan *framework* Laravel. Penyimpanan data menggunakan Relational Database MariaDB dan NoSQL Redis.

4.5.1 Implementasi Dataset

Implementasi *dataset* dimulai dengan melakukan *query* terhadap *database*. Proses *query* dilakukan dengan menggunakan metode ORM (*Object Relational Mapping*) sehingga tabel yang akan digunakan harus dituliskan dalam sebuah *class*. Dalam implementasi metode ORM, penulis menggunakan *library* bawaan dari *framework* Laravel yaitu Eloquent.

Berikut adalah daftar tabel beserta nama class ORM nya.

Tabel 4.1 Representasi tabel dalam *ORM Class*

No	Tabel	Nama *ORM Class*
1	mt_siswa	Student
2	mt_semester	Semester
3	mt_kompli	Major
4	bp_ref_point	Rule
5	mt_kelas	SchoolClass
6	bp_pelanggaran	Violation
7	jobs	Job

Student.php

```php
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Model;


class Student extends Model
{
    protected $connection = 'mysqlvsm';
    protected $table = 'mt_siswa';
    protected $primaryKey = 'nis';
    protected $appends = ['age'];

    public $timestamps = false;
```

Gambar 4.10 Kode Program Class Student

Gambar 4.10 adalah kode program untuk *class* dari tabel mt_siswa. Nama tabel dideklarasikan dalam properti $table pada baris 13. Kolom *primary key* dideklarasikan dalam properti $primaryKey pada baris 14.

Semester.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Semester extends Model
{
    protected $connection = 'mysqlvsm';
    protected $table = 'mt_semester';
    protected $primaryKey = 'smt_id';
}
```

Gambar 4.11 Kode Program Class Semester

Gambar 4.11 adalah kode program untuk *class* dari tabel mt_semester. Nama tabel di deklarasikan dalam properti $table pada baris 10. Kolom *primary key* di deklarasikan dalam properti $primaryKey pada baris 11.

Major.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Major extends Model
{
    protected $connection = 'mysqlvsm';
    protected $table = 'mt_kompli';
    protected $primaryKey = 'kompli_id';
}
```

Gambar 4.12 Kode Program Class Major

Gambar 4.12 adalah kode program untuk *class* dari tabel mt_kompli. Nama tabel di deklarasikan dalam properti $table pada baris 10. Kolom *primary key* di deklarasikan dalam properti $primaryKey pada baris 11.

Rule.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Rule extends Model
{
    protected $connection = 'mysqlvsm';
    protected $primaryKey = 'pnt_id';
    protected $table = 'bp_ref_point';
}
```

Gambar 4.13 Kode Program Class Rule

Gambar 4.12 adalah kode program untuk *class* dari tabel pnt_id_kompli. Nama tabel di deklarasikan dalam properti $table pada baris 10. Kolom *primary key* di deklarasikan dalam properti $primaryKey pada baris 11.

SchoolClass.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SchoolClass extends Model
{
    protected $connection = 'mysqlvsm';
    protected $table = 'mt_kelas';
    protected $primaryKey = 'kls_id';

}
```

Gambar 4.14 Kode Program Class SchoolClass

Gambar 4.14 adalah kode program untuk *class* dari tabel mt_kelas. Nama tabel di deklarasikan dalam properti $table pada baris 10. Kolom *primary key* di deklarasikan dalam properti $primaryKey pada baris 11.

Violation.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Violation extends Model
{
    protected $connection = 'mysqlvsm';
    protected $table = 'bp_pelanggaran';
    protected $primaryKey = 'pel_id';

}
```

Gambar 4.15 Kode Program Class Violation

Gambar 4.12 adalah kode program untuk *class* dari tabel bp_pelanggaran. Nama tabel di deklarasikan dalam properti $table pada baris 10. Kolom *primary key* di deklarasikan dalam properti $primaryKey pada baris 11.

Job.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Job extends Model
{

}
```

Gambar 4.16 Kode program class Job

Pada gambar 4.15, class Job tidak perlu memberikan properti $table dan $primaryKey karena secara *default* akan diartikan bahwa tabel yang digunakan bernama jobs dengan *primary key* kolom id.

Implementasi pembuatan *dataset* terdapat dalam *class* DatasetController *method* store. Pada *method* tersebut dilakukan *query* data siswa aktif di semester terakhir. Dari data hasil *query* tersebut dilakukan perulangan untuk membuat *dataset*

```php
public function store()
{
    $semester = Semester::orderBy('smt_id', 'desc')->first();
    $students = Student::activeStudentAt($semester->smt_id)->get();
    $dataset = $this->getDataset($students, $semester);
    DB::statement( query: 'SET FOREIGN_KEY_CHECKS=0;');
    DB::table( table: 'datasets')->truncate();
    $this->deleteCache();
    DB::table( table: 'datasets')->insert($dataset);
    DB::statement( query: 'SET FOREIGN_KEY_CHECKS=1;');
}

private function getDataset($students, $semester)
{
    $dataset = [];
    foreach ($students as $student) {
        $data['gender'] = $student->sis_jk;
        $data['junior_high_school'] = (is_null($student->sis_nama_asal_sltp)) ? '' : $student->sis_nama_asal_sltp;
        $data['older'] = $student->is_older;
        $data['major'] = $student->sis_kk_id;
        $data['parent_job'] = (is_null($student->sis_pek_ortu)) ? '' : $student->sis_pek_ortu;
        $data['class_label'] = $student->getClass($semester->smt_id);
        $dataset[] = $data;
    }
    return $dataset;
}
```

Gambar 4.17 Kode Program method store

Penjelasan dari kode program pada gambar 4.16:

Pada baris ke 35 terdapat variabel $semester yang melakukan *query* untuk mengambil data semester paling akhir. Pada baris 36 dilakukan *query* untuk mengambil data siswa yang aktif pada semester terpilih sesuai dengan variabel

$semester. Data siswa diproses menjadi *dataset* dengan memanggil *method* getDataset, lalu disimpan dalam variabel $dataset. Variabel $dataset disimpan dalam *database* label datasets.

Pada baris ke 45 terdapat *method* getDataset yang menerima parameter variabel $students dan variabel $semester. Data dalam variabel $students dilakukan perulangan dan diolah sesuai dengan kebutuhan atribut untuk *dataset*

```
67      public function scopeActiveStudentAt(Builder $query, $semester)
68      {
69          return $query->whereHas( relation: 'classHistories', function (Builder $query) use ($semester) {
70              $query->where( column: 'kls_ta', $semester);
71          });
72      }
73
74      public function getPoint($semester)
75      {
76          $total = 0;
77          $violations = $this->violations()->where( column: 'pel_smt_id', $semester)->get();
78          foreach ($violations as $violation) {
79              $total += $violation->pel_bobot;
80          }
81          return $total;
82      }
83
84      public function getClass($semester)
85      {
86          $point = $this->getPoint($semester);
87          return ($point > 30) ? 'undisciplined' : 'discipline';
88      }
```

Gambar 4.18 Kode active student at

Penjelasan dari kode program pada gambar 4.16:

Pada baris ke 67-72 terdapat *method* scopeActiveStudentAt yang digunakan untuk membuat *querybuilder* untuk memberikan filter semester saat *query* pengambilan siswa. Pada baris ke 74-82 terdapat *method* getPoint dengan parameter semester. *Method* ini digunakan untuk mengambil jumlah akumulasi poin siswa dalam 1 semeter. Pada baris ke 84-88 terdapat *method* getClass yang digunakan untuk memberikan label disiplin atau tidak disiplin terhadap siswa.

4.5.2 Implementasi Preprocessing Data

Dalam implementasi *preprocessing data* terdapat beberapa *method* yang digunakan, yaitu *method* untuk *missing value*, *convert*, dan *scaling*

```php
public function missingValue()
{
    $modus = Dataset::modus();
    DB::table( table: 'datasets')
        ->where( column: 'parent_job', operator: '')
        ->update(['parent_job' => $modus]);
}
```

Gambar 4.19 Kode Program method missingValue

```php
public function scopeModus($query)
{
    return $query->select('parent_job', DB::raw( value: 'count(*) as total'))
        ->groupBy('parent_job')
        ->orderBy('total', 'desc')->first()->parent_job;
}
```

Gambar 4.20 Kode program mendapatkan nilai modus

Penjelasan dari kode program pada gambar 4.18 dan gambar 4.19:

Dalam *method* missingValue() pada gambar 4.18, data yang tidak memiliki nilai akan digantikan dengan nilai modus. Nilai modus didapatkan dari *method* scopeModus yang terdapat dalam gambar 4.19 dengan melakukan proses *query groupingOutput* dari *query* tersebut selanjutnya diurutkan secara *descending* berdasarkan nilai total, lalu diambil data pertama sebagai nilai modus.

Dalam proses *convert data* penanganan tiap atribut berbeda-beda. Atribut jurusan tidak perlu diubah karena yang disimpan dalam basis data sudah berupa id yang bertipe data int. Atribut lebih tua disimpan dalam bentuk 0 dan 1. Atribut asal sekolah diubah menjadi urutan kolom asal SMP dalam tabel mt_siswa. Atribut pekerjaan orang tua disesuaikan dengan urutan pekerjaan berdasarkan id pekerjaan dalam tabel jobs.

```php
public function convertJuniorHighSchool()
{
    $list = Student::getJuniorHighSchool();
    foreach ($list as $key => $item) {
        DB::table( table: 'datasets')
            ->where( column: 'junior_high_school', $item->sis_nama_asal_sltp)
            ->update(['junior_high_school' => $key + 1]);
    }
}
```

Gambar 4.21 Kode Program method convertJuniorHighSchool

Penjelasan dari kode program pada gambar 4.20:

Pada baris 24 terdapat variabel $list yang mengambil nilai dari *method* getJuniorHighSchool. Variabel $list dilakukan perulangan untuk membuat *query updateQuery update* dilakukan dengan filter kolom junior_high_school yang bernilai sesuai dengan properti nama asal SMP pada setiap item $list.

```
90      public static function getJuniorHighSchool()
91      {
92          return Student::select('sis_nama_asal_sltp')
93              ->whereNotNull('sis_nama_asal_sltp')
94              ->where('sis_nama_asal_sltp', '≠', '')
95              ->orderBy('nis')->distinct()->get();
96      }
```

Gambar 4.22 Kode Program method getJuniorHighSchool

Gambar 4.21 adalah *method* yang digunakan untuk mendapatkan list asal SMP yang pernah terdaftar di SMK N 1 Pacitan. Pada baris 92 *method* mengembalikan nilai berupa *query select* sis_nama_asal_sltp dengan menambahkan *distinct* agar tidak terdapat duplikasi data.

```
52      public function convertParentJob()
53      {
54          $list = Job::all();
55          foreach ($list as $key ⇒ $item) {
56              DB::table( table: 'datasets')
57                  ->where( column: 'parent_job', $item->name)
58                  ->update(['parent_job' ⇒ $key + 1]);
59          }
60      }
```

Gambar 4.23 Kode Program method ConvertParentJob

Penjelasan dari kode program pada gambar 4.23:

Pada baris 54 adalah proses *query* untuk mengambil semua data pekerjaan dalam tabel jobs. Hasil *query* tersebut kemudian di lakukan perulangan untuk melakukan *query update* tabel *dataset* dengan kolom yang bernilai nama pekerjaan diubah menjadi urutan pekerjaan.

```
32      public function convertGender()
33      {
34          DB::table( table: 'datasets')
35              ->where( column: 'gender', operator: 'L')
36              ->update(['gender' => '1']);
37          DB::table( table: 'datasets')
38              ->where( column: 'gender', operator: 'P')
39              ->update(['gender' => '0']);
40      }
```

Gambar 4.24 Kode program convert gender

Penjelasan dari kode program pada gambar 4.24:

Pada *method* convertGender terdapat 2 proses *query*. *Query* pertama adalah *query* untuk mengubah data dengan kolom gender bernilai L menjadi 1 yang terdapat dalam baris 52-54. *Query* kedua untuk mengubah data dengan kolom gender bernilai P menjadi 0 yang terdapat pada baris 55-57.

Untuk *class labeling class discipline* diubah menjadi 0 dan class undisciplined menjadi 1. Proses tersebut dilakukan di dalam method convertClass().

```
42      public function convertClass()
43      {
44          DB::table( table: 'datasets')
45              ->where( column: 'class_label', operator: 'discipline')
46              ->update(['class_label' => '0']);
47          DB::table( table: 'datasets')
48              ->where( column: 'class_label', operator: 'undisciplined')
49              ->update(['class_label' => '1']);
50      }
```

Gambar 4.25 Kode Program method convert class

Pada baris 44 terdapat *query* untuk melakukan *update* data dengan class_label bernilai discipline menjadi 0. Pada baris 47 terdapat query untuk melakukan *update* data dengan class_label bernilai *undisciplined* menjadi 1.

```php
public function scalingInDB()
{
    $min = $this->getMinAtrributeValue();
    $max = $this->getMaxAtrributeValue();
    DB::table( table: 'datasets')
        ->update([
            'junior_high_school' => DB::raw( value: "
                ((junior_high_school+ 0.0) - " . $min['juniorHighSchool'] . ") /
                (" . $max['juniorHighSchool'] . "-" . $min['juniorHighSchool'] . ")
            "),
            'major' => DB::raw( value: "
                ((major+ 0.0) - " . $min['major'] . ") /
                (" . $max['major'] . "-" . $min['major'] . ")
            "),
            'parent_job' => DB::raw( value: "
                ((parent_job+ 0.0) - " . $min['parentJob'] . ") /
                (" . $max['parentJob'] . "-" . $min['parentJob'] . ")
            "),
        ]);
}
```

Gambar 4.26 Kode Program method scaling

Pada baris 33 dalam gambar 4.26 terdapat variabel $min yang memanggil *method* getMinAttribute untuk mendapatkan nilai minimal dari masing-masing atribut. Pada baris 34 terdapat variabel $max yang memanggil *method* getMaxAttribute untuk mendapatkan nilai maksimal dari masing-masing atribut. Baris 35-48 adalah proses *query* untuk *update* semua atribut.

```php
public function getMinAttributeValue()
{
    $min = Redis::get('min');
    if ($min == null) {
        $min = (Object)[
            'major' => Major::min('kompli_id'),
            'juniorHighSchool' => 1,
            'parentJob' => Job::min('id')
        ];
        Redis::set('min', json_encode($min));
    } else {
        $min = json_decode($min);
    }
    return $min;
}
```

Gambar 4.27 Kode program get min atribut

Pada baris 64 dalam gambar 4.27 terdapat variabel $min yang mengambil data dari *cache* Variabel $min dilakukan pengecekan pada baris 65, jika nilai variabel $min kosong maka dilakukan pengambilan data dari *database*

```php
78      public function getMaxAttributeValue()
79      {
80          $max = Redis::get('max');
81          if ($max == null) {
82              $max = (Object)[
83                  'major' => Major::max('kompli_id'),
84                  'juniorHighSchool' => Student::getJuniorHighSchool()->count(),
85                  'parentJob' => Job::max('id')
86              ];
87              Redis::set('max', json_encode($max));
88          } else {
89              $max = json_decode($max);
90          }
91          return $max;
92      }
```

Gambar 4.28 Kode program get max

Pada baris 65 dalam gambar 4.28 terdapat variabel $max yang mengambil data dari *cache*. Variabel $max dilakukan pengecekan pada baris 81, jika nilai variabel $max adalah *null* maka dilakukan pengambilan data dari *database*

4.5.3 Implementasi Training

Implementasi *training* dilakukan dalam class NeuralNetwork. Class NeuralNetwork dapat dilihat dalam gambar berikut:

```php
11  class NeuralNetwork
12  {
13      private $input = [];
14
15      private $output = [];
16
17      private $firstLayerWeights;
18
19      private $secondLayerWeights;
20
21      private $hiddenNeuron = 3;
22
23      private $outputNeuron = 1;
24
25      private $secondLayerSigmoid;
26      private $secondLayerSum;
27      private $firstLayerSigmoid;
28      private $firstLayerSum;
29
30      private $dJdW2;
31      private $dJdW1;
32
33      private $learningRate = 0.01;
34
35      private $k;
36
37      private $datasetRepository;
38
```

Gambar 4.29 Kode Program NeuralNetwork.php

Gambar 4.30 menunjukkan seluruh properti yang digunakan dalam kelas Neural Network. Penjelasan properti dari *class* tersebut dapat dilihat dalam tabel berikut:

Tabel 4.2 Deskripsi property class Neural Network

Properti	Deskripsi
$input	Data *input*
$output	Data *output*
$firstLayerWeights	Bobot pada *input layer* menuju *hidden layer*
$secondLayerWeights	Bobot pada *hidden layer* menuju *output layer*
$hiddenNeuron	Jumlah neuron pada *hidden* layer
$outputNeuron	Jumlah neuron pada *output layer*
$secondLayerSigmoid	Hasil fungsi akivasi (*sigmoid*) pada *output layer*
$secondLayerSum	Hasil penjumlahan *sigmoid* pada *output layer*
$firstLayerSigmoid	Hasil fungsi aktivasi (*sigmoid*) pada hidden layer
$firstLayerSum	Hasil penjumlahan *sigmoid* pada hidden layer
$dJdW2	Nilai turunan dari output layer terhadap hidden layer
$dJdW1	Nilai turunan dari output layer terhadap input layer
$learningRate	Nilai yang digunakan untuk melakukan perbaruan bobot

Pemberian nilai *input* dilakukan di dalam *method* getInput(). Sedangkan untuk *output* di dalam *method* getOutput(). Data *input* dan *output* diambil dari *cache*. Saat *cache* kosong data diambil dari *database*

```
42      public function __construct(DatasetRepository $datasetRepository, $k, $test = false)
43      {
44          $this->datasetRepository = $datasetRepository;
45          $this->k = $k;
46          if (!$test) {
47              $k = Redis::get('k');
48              if ($k == null || $k != $this->k) {
49                  $datasetRepository->removeCache();
50                  Redis::set('k', $this->k);
51              }
52              $this->getInput();
53              $this->getOutput();
54              $this->createRandomWeights();
55          }
56      }
57
58      public function setInput($input)
59      {
60          $this->input = $this->addBias($input);
61      }
62
63      public function setOutput($output)
64      {
65          $this->output = $output;
66      }
```

Gambar 4.30 Potongan kode program neural network

Penjelasan dari kode program pada gambar 4.31:

Baris 42-56 adalah *constructor* yang dijalankan saat *class* di *instantiate*. *Constructor* digunakan untuk pemberian nilai awal. Di dalam *constructor* memanggil *method* lain, yaitu getInput, getOutput, dan createRandomWeight. Pada baris 58-61 terdapat kode program yang berfungsi untuk memberikan nilai input. Baris 63-66 terdapat kode program yang berfungsi untuk memberikan nilai output.

```
68      public function setFirstWeight($weight)
69      {
70          $this->firstLayerWeights = $weight;
71      }
72
73      public function setSecondWeight($weight)
74      {
75          $this->secondLayerWeights = $weight;
76      }
77
78      public function getInput()
79      {
80          $this->input = $this->addBias($this->datasetRepository->getInput($this->k));
81      }
82
83      public function getOutput()
84      {
85          $this->output = $this->datasetRepository->getOutput($this->k);
86      }
87
88      private function addBias(Array $a)
89      {
90          for ($i = 0; $i < count($a); $i++) {
91              $a[$i][count($a[$i])] = 1;
92          }
93          return $a;
94      }
```

Gambar 4.31 Potongan kode program neural network

Penjelasan dari kode program pada gambar 4.31:

Baris 68-71 adalah kode program untuk memberikan nilai bobot pada layer pertama. Pemberian nilai bobot pada layer kedua terdapat pada baris 73-76. Pada baris 78-81 adalah *method* untuk mendapatkan data *input* dari *database*. Pada baris 83-86 adalah *method* untuk mendapatkan data *output* dari *database*. Pada baris 88-94 terdapat *method* addBias yang berfungsi untuk menambahkan nilai 1 di akhir data input.

```php
public function getInput($k)
{
    $x = Redis::get('input');
    if ($x == null) {
        $input = $this->getDatabaseInput($k)->toArray();
        Redis::set('input', json_encode($input));
    } else {
        $input = json_decode($x);
    }
    return $input;
}

private function getDatabaseInput($k)
{
    return $this->getDataset($k, status: 'train')->map(function ($item) {
        return [
            (double)$item->gender,
            (double)$item->junior_high_school,
            (double)$item->older,
            (double)$item->major,
            (double)$item->parent_job,
        ];
    });
}
```

Gambar 4.32 Kode program getInput

Penjelasan dari kode program pada gambar 4.33:

Pada baris 175-185 terdapat *method* getInput dengan parameter $k yang digunakan untuk mendapatkan *input* sesuai dengan k yang diinginkan. Pada baris 177 terdapat variabel $x yang mengambil nilai *input* dari *cache*. Nilai dari variabel $x dilakukan pengecekan pada baris 178, Jika bernilai kosong maka akan dilakukan pengambilan data dari *database*. Pengambilan dilakukan dengan memanggil method getDatabaseInput yang terdapat pada baris 187-198.

```php
public function getOutput($k)
{
    $x = Redis::get('output');
    if ($x == null) {
        $output = $this->getDatabaseOutput($k);
        Redis::set('output', json_encode($output));
    } else {
        $output = json_decode($x);
    }
    return $output;
}

private function getDatabaseOutput($k)
{
    return $this->getDataset($k, status: 'train')->map(function ($item) {
        return [
            (double)$item->class_label,
        ];
    });
}
```

Gambar 4.33 Kode program get output

Penjelasan dari kode program pada gambar 4.34:

Pada baris 213-223 terdapat *method* getOutput dengan parameter $k yang digunakan untuk mendapatkan *output* sesuai dengan k yang diinginkan. Pada baris 215 terdapat variabel $x yang mengambil nilai *output* dari *cache*. Nilai dari variabel $x dilakukan pengecekan pada baris 216, Jika bernilai kosong maka akan dilakukan pengambilan data dari *database*. Pengambilan dilakukan dengan memanggil *method* getDatabaseOutput yang terdapat pada baris 225-232.

Pemberian nilai bobot awal dilakukan dengan mengambil nilai acak antara 0 - 1. Selanjutnya nilai bobot akan disimpan dalam *cache*.

```
96      private function createRandomWeights()
97      {
98          $this->createFirstLayerWeight();
99          $this->createSecondLayerWeight();
100     }
```

Gambar 4.34 Kode program pembuatan bobot acak

Gambar 4.35 adalah proses pembuatan bobot acak. Proses tersebut dipisah dalam 2 sub proses, yaitu bobot layer pertama dan bobot layer kedua.

```
102     private function createFirstLayerWeight()
103     {
104         $x = Redis::get('firstLayerWeights');
105         if ($x == null) {
106             $this->firstLayerWeights = [];
107             for ($i = 0; $i < count($this->input[0]); $i++) {
108                 $w = [];
109                 for ($j = 0; $j < $this->hiddenNeuron; $j++) {
110                     $w[] = rand(0, 100) / 100;
111                 }
112                 $this->firstLayerWeights[] = $w;
113             }
114         } else {
115             $this->firstLayerWeights = json_decode($x);
116         }
117     }
```

Gambar 4.35 Kode program pembuatan bobot *input* layer

Gambar 4.36 adalah proses pemberian bobot untuk layer pertama. Pada baris 104 terdapat kode untuk mengambil bobot dari *cache* yang disimpan dalam

variabel $x. Pada baris 105 dilakukan pengecekan terhadap variabel $x, jika nilai variabel $x adalah *null* maka akan dilakukan pemberian bobot secara acak.

```
119     private function createSecondLayerWeight()
120     {
121         $x = Redis::get('secondLayerWeights');
122         if ($x == null) {
123             $this->secondLayerWeights = [];
124             for ($i = 0; $i <= $this->hiddenNeuron; $i++) {
125                 $w = [];
126                 for ($j = 0; $j < $this->outputNeuron; $j++) {
127                     $w[] = rand(-100, 100) / 100;
128                 }
129                 $this->secondLayerWeights[] = $w;
130             }
131         } else {
132             $this->secondLayerWeights = json_decode($x);
133         }
134     }
```

Gambar 4.36 Kode program pembuatan bobot *hidden layer*

Gambar 4.37 adalah proses pemberian bobot untuk layer ke dua. Pada baris 121 terdapat kode untuk mengambil bobot dari *cache* yang disimpan dalam variabel $x. Pada baris 122 dilakukan pengecekan terhadap variabel $x, jika nilai variabel $x adalah *null* maka akan dilakukan pemberian bobot secara acak.

Proses *training* terdapat dalam *method* train(). Perulangan yang dilakukan dalam proses *training* adalah 1000. Terdapat 4 proses dalam proses *training*. Proses pertama adalah feed forward. Proses ini melakukan perhitungan input dan bobot pada setiap layer. Selanjutnya adalah menghitung nilai selisih dari output sebenarnya dan *output* hasil *feed forward*. Dari selisih tersebut kemudian dihitung mundur untuk mencari nilai turunannya. Setelah itu *update* nilai bobot sesuai dengan nilai *learning rate* dan nilai turunannya.

```
223     public function train($epoch)
224     {
225         for ($i = 0; $i < $epoch; $i++) {
226             echo $i . "\n";
227             if (($i + 1) % floor( value: $epoch / 10) == 0 || $i == ($epoch+1))
228                 $this->notif($i, $epoch);
229             $this->feedForward();
230             $this->costFunction();
231             $this->costFunctionPrime();
232             $this->updateWeight();
233         }
234         $this->savePredict();
235         $this->saveWeight();
236         Redis::set('isTraining', false);
237     }
```

Gambar 3.2 Kode program proses training

Pada baris 225 – 233 adalah proses perulangan sebanyak *epoch* yang ditentukan. Di dalam perulangan terdapat proses notifikasi ke *user* pada baris 228 agar *user* dapat memantau proses *training*. Setelah perulangan selesai dilakukan penyimpanan hasil *predict* dan bobot terakhir pada baris 234 dan 235.

```
239     private function notif($i, $epoch)
240     {
241         $prosentase = (($i + 1) / $epoch) * 100;
242         event(new TrainingFinished($prosentase));
243     }
```

Gambar 3.3 Kode program notif

Gambar 3.3 adalah proses untuk notifikasi progres proses *training* terhadap *user*. Progres dikirim dalam bentuk persentase *epoch*

```
136     public function feedForward()
137     {
138         try {
139             $this->firstLayerSum = Num::dot($this->input, $this->firstLayerWeights);
140             $this->firstLayerSigmoid = $this->addBias($this->sigmoid($this->firstLayerSum));
141             $this->secondLayerSum = Num::dot($this->firstLayerSigmoid, $this->secondLayerWeights);
142             $this->secondLayerSigmoid = $this->sigmoid($this->secondLayerSum);
143         } catch (Exception $e) {
144             echo $e->getMessage();
145         }
146         return $this->secondLayerSigmoid;
147     }
```

Gambar 4.37 Kode program proses feed forward

Pada gambar 4.38 terdapat empat tahapan dalam *method* feedForward(). Tahapan pertama adalah perhitungan *input* dengan bobot layer pertama yang terdapat dalam baris 110. Tahapan kedua adalah proses perhitungan nilai *sigmoid* dalam *hidden layer* pada baris 111.

```
160     public function sigmoid($z)
161     {
162         for ($i = 0; $i < count($z); $i++) {
163             for ($j = 0; $j < count($z[$i]); $j++) {
164                 $z[$i][$j] = 1 / (1 + exp( arg: -1 * $z[$i][$j]));
165             }
166         }
167         return $z;
168     }
```

Gambar 4.38 Kode program proses feed forward

Pada gambar 4.39 terdapat implementasi perhitungan dari rumus fungsi *sigmoid*

```
149     public function costFunction()
150     {
151         $costFunction = 0;
152         for ($i = 0; $i < count($this->output); $i++) {
153             for ($j = 0; $j < count($this->output[$i]); $j++) {
154                 $costFunction += ($this->output[$i][$j] - $this->secondLayerSigmoid[$i][$j]) ** 2 / 2;
155             }
156         }
157         return $costFunction;
158     }
```

Gambar 4.39 Kode program proses perhitungan *cost function*

Pada gambar 4.40 terdapat *method* costFunction. *Method* costFunction digunakan untuk menghitung nilai selisih antara *output* yang didapat dengan nilai *output* yang diinginkan.

```
170     public function costFunctionPrime()
171     {
172         $delta3 = Num::multipleScalar( a: -1, Num::minus($this->output, $this->secondLayerSigmoid));
173         $delta3 = Num::multiply($delta3, $this->sigmoidPrime($this->secondLayerSum));
174         $this->dJdW2 = Num::dot(Num::transpose($this->firstLayerSigmoid), $delta3);
175         $delta2 = Num::dot($delta3, Num::transpose($this->secondLayerWeights));
176         $delta2 = Num::multiply($delta2, $this->sigmoidPrime($this->addBias($this->firstLayerSum)));
177         $this->dJdW1 = Num::dot(Num::transpose($this->input), $delta2);
178         return [$this->dJdW1, $this->dJdW2];
179     }
180
181     public function sigmoidPrime($z)
182     {
183         for ($i = 0; $i < count($z); $i++) {
184             for ($j = 0; $j < count($z[$i]); $j++) {
185                 $z[$i][$j] = exp( arg: -1 * $z[$i][$j]) / ((1 + exp( arg: -1 * $z[$i][$j])) ** 2);
186             }
187         }
188         return $z;
189     }
```

Gambar 4.40 Kode program proses perhitungan mundur

Pada gambar 4.42 terdapat 2 *method* *Method* pertama adalah *method* perhitungan mundur untuk mendapatkan nilai turunan yang nantinya digunakan untuk memperbarui bobot. Pada baris 172 - 174 adalah proses perhitungan mundur

dari *output layer* ke *hidden layer*. Pada baris 175 - 177 adalah proses perhitungan mundur dari *hidden layer* ke *input layer*. *Method* kedua adalah *method* yang digunakan untuk mendapat nilai turunan dari fungsi *sigmoid*.

```
191     public function updateWeight()
192     {
193         $this->updateSecondLayerWeight();
194         $this->updateFirstLayerWeight();
195     }
196
197     private function updateFirstLayerWeight()
198     {
199         for ($i = 0; $i < count($this->firstLayerWeights); $i++) {
200             for ($j = 0; $j < count($this->firstLayerWeights[$i]); $j++) {
201                 $this->firstLayerWeights[$i][$j] =
202                     $this->firstLayerWeights[$i][$j] - $this->learningRate * $this->dJdW1[$i][$j];
203             }
204         }
205     }
206
207     private function updateSecondLayerWeight()
208     {
209         for ($i = 0; $i < count($this->secondLayerWeights); $i++) {
210             for ($j = 0; $j < count($this->secondLayerWeights[$i]); $j++) {
211                 $this->secondLayerWeights[$i][$j] =
212                     $this->secondLayerWeights[$i][$j] - $this->learningRate * $this->dJdW2[$i][$j];
213             }
214         }
215     }
```

Gambar 4.41 Kode program proses pembaruan bobot

Pada gambar 4.42 terdapat 3 *method*. *Method* baris 191-195 adalah *method* untuk melakukan *update* seluruh bobot, *method* ini memanggil *method* updateSecondLayerWeight dan *method* updateFirstLayerWeight. *Method* pada baris 167-175 adalah *method* untuk melakukan *update* bobot pada layer pertama dengan menggunakan variabel *learning rate* dan variabel $dJdW1. *Method* baris 177-185 adalah *method* untuk melakukan *update* bobot pada layer kedua dengan menggunakan variabel *learning rate* dan variabel $dJdW2.

```
245     private function savePredict()
246     {
247         if ($this->k > 0) {
248             $this->input = $this->addBias($this->datasetRepository->getTestInput($this->k)->toArray());
249             $this->output = $this->addBias($this->datasetRepository->getTestOutput($this->k)->toArray());
250         }
251         $this->feedForward();
252         $collection = collect($this->secondLayerSigmoid);
253         $predict = $collection->map(function ($item) {
254             $x = [];
255             foreach ($item as $i) {
256                 $x[] = ($i < 0.5) ? 0 : 1;
257             }
258             return $x;
259         });
260         Redis::set('predict', json_encode($predict));
261     }
262 }
```

Gambar 4.42 Kode program penyimpanan hasil prediksi

Gambar 4.42 adalah penyimpanan hasil prediksi ke dalam cache. Pada baris ke 247 terdapat pengecekan nilai $k. Jika nilai $k lebih dari 0 maka data input dan output diambil ulang dari database sesuai dengan nilai $k.

4.5.4 Implementasi Resampling

```
115    public function overSampling()
116    {
117        $undisciplined = Dataset::undisciplined()->get();
118        $discipline = Dataset::discipline()->count();
119        $diff = $discipline - $undisciplined->count();
120        for ($i = 0; $i < $diff; $i++) {
121            Dataset::create($undisciplined[rand(0, $undisciplined->count() - 1)]->toArray());
122        }
123    }
```

Gambar 4.43 Kode program over sampling

Gambar 4.43 adalah kode program untuk melakukan *oversampling*. Baris 117 adalah proses *query* untuk mengambil seluruh data tidak disiplin sedangkan baris 118 proses *query* untuk mengambil seluruh data disiplin. Baris 119 menghitung selisih antara data disiplin dan tidak disiplin. Baris 120-123 proses mengambil data tidak disiplin secara acak untuk disimpan kembali dalam *dataset* sebanyak selisih data.

```
125    public function underSampling()
126    {
127        $undisciplined = Dataset::undisciplined()->count();
128        $discipline = Dataset::discipline()->inRandomOrder()->get();
129        for ($i = $undisciplined; $i < $discipline->count(); $i++) {
130            $discipline[$i]->delete();
131        }
132    }
```

Gambar 4.44 Kode program under sampling

Gambar 4.44 adalah kode program untuk melakukan *undersampling*. Baris 117 adalah proses *query* untuk mengambil seluruh data tidak disiplin. Baris 118 proses *query* untuk mengambil seluruh data disiplin dengan urutan secara acak. Baris 129-131 proses penghapusan data disiplin sebanyak selisih data.

4.5.5 Implementasi Pembagian Data

```
134     public function split()
135     {
136         $discipline = Dataset::discipline()->inRandomOrder()->get();
137         $undisciplined = Dataset::undisciplined()->inRandomOrder()->get();
138         $this->splitDataset($discipline);
139         $this->splitDataset($undisciplined);
140     }
141
142     private function splitDataset($data)
143     {
144         $pos = 1;
145         foreach ($data as $datum) {
146             $datum->update(['position' => $pos++]);
147             $pos = ($pos == 11) ? 1 : $pos;
148         }
149     }
```

Gambar 4.45 Kode program split

Gambar 4.45 adalah rangkaian proses *split* data. Pada baris 136 dilakukan proses *query* untuk pengambilan data disiplin dengan urutan secara acak sedangkan baris 137 untuk pengambilan data tidak disiplin secara acak. Kedua data tersebut kemudian dilakukan pemberian label k untuk *split* dengan memanggil *method* splitDataset pada baris 138 dan baris 139. Dalam *method* splitDataset, nilai k diperbarui berdasarkan nilai variabel $pos. Nilai variabel $pos bertambah 1 setiap proses perulangan dan direset menjadi 0 setelah mencapai nilai 11.

4.6 Pengujian Sistem

Pengujian dilakukan untuk mengevaluasi sistem yang dibuat telah sesuai dengan yang diharapkan. Pengujian dilakukan menggunakan PHP Unit.

Tabel 4.3 Pengujian class NumPHP

No	Method	Deskripsi	Hasil
1	dot	*Method* untuk melakukan perkalian 2 matrik	OK
2	multiply	*Method* untuk melakukan perkalian 2 array	OK
3	transpose	*Method* untuk melakukan *transpose* sebuah matrik	OK
4	multipleScalar	*Method* untuk melakukan perkalian matrik dengan sebuah bilangan	OK

```php
public function testNumDot()
{
    $a = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ];
    $b = [
        [6, 5],
        [4, 3],
        [2, 1]
    ];
    $result = [
        [20, 14],
        [56, 41],
        [92, 68]
    ];
    $this->assertTrue( condition: $result == Num::dot($a, $b));
}
```

Gambar 4.46 Kode program pengetesan fungsi perkalian matrik

Pada gambar 4.46 terdapat 3 variabel *array* dengan nama $a, $b, dan $result. Pengetesan dilakukan dengan cara memasukkan variabel $a dan $b ke dalam parameter *method* dot. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* dot sama dengan nilai variabel $result.

```php
public function testNumMultiply()
{
    $a = [
        [1, 2],
        [1, 2]
    ];
    $b = [
        [1, 2],
        [1, 2]
    ];
    $result = [[1, 4], [1, 4]];
    $this->assertTrue( condition: $result == Num::multiply($a, $b));
}
```

Gambar 4.47 Kode program pengetesan fungsi perkalian array

Pada gambar 4.47 terdapat 3 variabel *array* dengan nama $a, $b, dan $result. Pengetesan dilakukan dengan cara memasukkan variabel $a dan $b ke dalam parameter *method* multiply. hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* multiply sama dengan nilai variabel $result.

```php
public function testTranspose()
{
    $a = [
        [1, 2],
        [4, 5],
        [7, 8]
    ];
    $result = [
        [1, 4, 7],
        [2, 5, 8]
    ];
    $this->assertTrue( condition: $result == Num::transpose($a));
}
```

Gambar 4.48 Kode program pengetesan fungsi transpose

Pada gambar 4.48 terdapat 2 variabel *array* dengan nama $a dan $result. Pengetesan dilakukan dengan cara memasukkan variabel $a ke dalam parameter *method* transpose. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* transpose sama dengan nilai variabel $result.

```php
public function testMultipleScalar()
{
    $a = 3;
    $b = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ];
    $result = [
        [3, 6, 9],
        [12, 15, 18],
        [21, 24, 27]
    ];
    $this->assertTrue( condition: $result == Num::multipleScalar($a,$b));
}
```

Gambar 4.49 Kode program pengetesan fungsi multipleScalar

Pada gambar 4.49 terdapat 2 variabel *array* dengan nama $a, $b, dan $result. Pengetesan dilakukan dengan cara memasukkan variabel $a dan $b ke dalam parameter *method* multipleScalar. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* multipleScalar sama dengan nilai variabel $result.

Tabel 4.4 Pengujian class NeuralNetwork

No	Method	Deskripsi	Hasil
1	sigmoid	Method untuk melakukan fungsi aktivasi	OK
2	feedForward	Method untuk melakukan perhitungan input ke output	OK
3	costFunction	Method untuk melakukan perhitungan cost selisih prediksi dan nilai asli	OK
4	costFunctionPrime	Method untuk menghitung gradient descent	OK

```
45    public function testSigmoid()
46    {
47        $x = [[2]];
48        $result = [[0.88079707797788]];
49        $this->assertEquals($result, $this->neuralNetwork->sigmoid($x));
50    }
```

Gambar 4.50 Kode program pengetesan *method* sigmoid

Pada gambar 4.50 terdapat 2 variabel *array* dengan nama $x dan $result. Pengetesan dilakukan dengan cara memasukkan variabel $x ke dalam parameter *method* sigmoid. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* sigmoid sama dengan nilai variabel $result.

```
52    public function testFeedForward()
53    {
54        $result = [
55            [ ... ],
58            [ ... ],
61            [ ... ],
64        ];
65        $this->assertEquals($result, $this->neuralNetwork->feedForward());
66    }
```

Gambar 4.51 Kode program pengetesan *method* feedForward

Pada gambar 4.51 terdapat variabel *array* dengan nama $result. Pengetesan dilakukan dengan cara mengeksekusi *method* feedForward. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* feedForward sama dengan nilai variabel $result.

```
71    public function testCostFunction()
72    {
73        $result = 0.47644455071418;
74        $this->neuralNetwork->feedForward();
75        $this->assertEquals($result, $this->neuralNetwork->costFunction());
76    }
```

Gambar 4.52 Kode program pengetesan *method* costFunction

Pada gambar 4.52 terdapat variabel *array* dengan nama $result. Pengetesan dilakukan dengan cara mengeksekusi *method* feedForward lalu *method* costFunction. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* costFunction sama dengan nilai variabel $result.

```
75    public function testCostFunctionPrime()
76    {
77        $result = [ ... ];
102       $this->neuralNetwork->feedForward();
103       $this->neuralNetwork->costFunction();
104       $this->assertEquals($result, $this->neuralNetwork->costFunctionPrime());
105   }
```

Gambar 4.53 Kode program pengetesan *method* costFunctionPrime

Pada gambar 4.53 terdapat variabel *array* dengan nama $result. Pengetesan dilakukan dengan cara mengeksekusi *method* feedForward, *method* costFunction, lalu *method* costFunctionPrime. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* costFunctionPrime sama dengan nilai variabel $result.

```
121   public function testUpdateSecondLayerWeight()
122   {
123       $result = [ ... ];
137       $this->neuralNetwork->feedForward();
138       $this->neuralNetwork->costFunction();
139       $this->neuralNetwork->costFunctionPrime();
140       $this->assertEquals($result, $this->neuralNetwork->updateSecondLayerWeight());
141   }
```

Gambar 4.54 Kode program pengetesan *method* updateSecondLayerWeight

Pada gambar 4.54 terdapat variabel *array* dengan nama $result. Pengetesan dilakukan dengan cara mengeksekusi *method* feedForward, *method* costFunction, ,*method* costFunctionPrime, kemudian *method* updateSecondLayerWeight. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan

berhasil jika nilai kembalian dari *method* updateSecondLayerWeight sama dengan nilai variabel $result.

```
143 ▶   public function testUpdateFirstLayerWeight()
144     {
145         $result = [ ... ];
162         $this→neuralNetwork→feedForward();
163         $this→neuralNetwork→costFunction();
164         $this→neuralNetwork→costFunctionPrime();
165         $this→assertEquals($result, $this→neuralNetwork→updateFirstLayerWeight());
166     }
```

Gambar 4.55 Kode program pengetesan *method* updateSecondLayerWeight

Pada gambar 4.55 terdapat variabel *array* dengan nama $result. Pengetesan dilakukan dengan cara mengeksekusi *method* feedForward, *method* costFunction, ,*method* costFunctionPrime, kemudian *method* updateFirstLayerWeight. Hasilnya akan dibandingkan dengan nilai variabel $result. Pengetesan dinyatakan berhasil jika nilai kembalian dari *method* updateFirstLayerWeight sama dengan nilai variabel $result.

BAB 6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan analisis dan pengujian yang dilakukan pada bab sebelumnya, maka kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Penerapan neural network dilakukan dengan mengubah data bertipe input yang bernilai string menjadi numerik. Proses pencarian bobot terbaik dilakukan dengan menggunakan *backpropagation*.
2. Sistem prediksi ketidakdisiplinan siswa ini dapat mengklasifikasikan siswa berdasarkan data diri siswa dengan *F-Measure* yang cukup baik. Sistem mampu mengklasifikasikan ke dalam dua jenis siswa, yaitu disiplin dan tidak disiplin.
3. Berdasarkan hasil pengujian, model yang dihasilkan memperoleh nilai rata-rata *F-Measure* sebesar 76.70% pada saat menggunakan *random over sampling* dan 76.29% pada saat menggunakan *random under sampling*.
4. Keseimbangan rasio dataset dapat mempengaruhi nilai *F-Measure*. *Dataset* yang tidak seimbang pada penelitian melewati proses *resampling* agar performa yang dihasilkan dapat menjadi lebih maksimal.

6.2 Saran

Saran yang dapat diajukan berdasarkan hasil penelitian adalah:

1. Perlu dilakukan pemilihan atribut yang lebih berkorelasi dengan ketidakdisiplinan siswa. Atribut yang didapat dari sistem informasi manajemen masih kurang lengkap dan banyak *noise* data.
2. Dalam penanganan ketidakseimbangan kelas, proses *resampling* perlu dicoba dengan algoritma lain sehingga mampu memilih data mana yang digunakan

DAFTAR PUSTAKA

Arisana, A. L., & Ismani. (2012). PENGARUH KEDISIPLINAN SISWA DAN PERSEPSI SISWA TENTANG KUALITAS MENGAJAR GURU TERHADAP PRESTASI BELAJAR AKUNTANSI SISWA KELAS XI IPS MAN YOGYAKARTA II TAHUN AJARAN 2011/2012. *Jurnal Pendidikan Akuntansi Indonesia*, 2-13.

Astuti, N. P., Kusrini, & Arief, M. R. (2015). ANALISIS PREDIKSI TINGKAT KETIDAKDIPLINAN SISWA MENGGUNAKAN ALGORITMA NAÏVE BAYES CLASSIFIER (STUDI KASUS : SMK NEGERI 1 PACITAN). *Seminar Nasional Teknologi Informasi dan Multimedia*, 91-96.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques Third Edition*. Waltham: Elsevier Inc.

Muschkin, C. G. (2014, Maret 11). *Students who repeat a year stoke bad behaviour in class* Retrieved from The Conversation: https://theconversation.com/students-who-repeat-a-year-stoke-bad-behaviour-in-class-23956

Nurhayati, Soekarno, I., & Hadihardaja, I. K. (2014). A Study of Hold-Out and K-Fold Cross Validation for Accuracy of Groundwater Modeling in Tidal Lowland Reclamation Using Extreme Learning Machine. *International Conference on Technology, Informatics, Management, Engineering & Environment*, 228-233.

Saifudin, A., & Wahono, R. S. (2015). Pendekatan Level Data untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software. *Journal of Software Engineering*, 76-85.

Santoso, B., & Umam, A. (2018). *Data Mining dan Big Data Analytics.* Yogyakarta: Penebar Media Pustaka.

Setiawan, D., Putri, R. N., & Suryanita, R. (2019). Perbandingan Algoritma Genetika dan Backpropagation pada Aplikasi Prediksi Penyakit Autoimun. *Khazanah Informatika*, *5*(2), 21-27.

Sharma, A. K., Prajapat, S. K., & Aslam, M. (2014). A Comparative Study between Naïve Bayes and Neural Network (MLP) Classifier for Spam Email Detection. *International Journal of Computer Applications*, *12*(16).

Soares, F. M., & Souza, A. M. (2016). *Neural Network Programming with Java.* Birmingham: Packt Publishing.

Suyanto. (2018). *MachineLearningTingkatDasar dan Lanjut.* Bandung: Informatika.

LAMPIRAN

Daftar data pekerjaan

BELUM/TIDAK BEKERJA	1
MENGURUS RUMAH TANGGA	2
PELAJAR/MAHASISWA	3
PENSIUNAN	4
PEGAWAI NEGERI SIPIL	5
TENTARA NASIONAL INDONESIA	6
KEPOLISIAN RI	7
PERDAGANGAN	8
PETANI/PEKEBUN	9
PETERNAK	10
NELAYAN/PERIKANAN	11
INDUSTRI	12
KONSTRUKSI	13
TRANSPORTASI	14
KARYAWAN SWASTA	15
KARYAWAN BUMN	16
KARYAWAN BUMD	17
KARYAWAN HONORER	18
BURUH HARIAN LEPAS	19
BURUH TANI/PERKEBUNAN	20
BURUH NELAYAN/PERIKANAN	21
BURUH PETERNAKAN	22
PEMBANTU RUMAH TANGGA	23
TUKANG CUKUR	24

TUKANG LISTRIK	25
TUKANG BATU	26
TUKANG KAYU	27
TUKANG SOL SEPATU	28
TUKANG LAS/PANDAI BESI	29
TUKANG JAHIT	30
TUKANG GIGI	31
PENATA RIAS	32
PENATA BUSANA	33
PENATA RAMBUT	34
MEKANIK	35
SENIMAN	36
TABIB	37
PARAJI	38
PERANCANG BUSANA	39
PENTERJEMAH	40
IMAM MESJID	41
PENDETA	42
PASTOR	43
WARTAWAN	44
USTADZ/MUBALIGH	45
JURU MASAK	46
PROMOTOR ACARA	47
ANGGOTA DPR-RI	48
ANGGOTA DPD	49
ANGGOTA BPK	50

PRESIDEN	51
WAKIL PRESIDEN	52
ANGGOTA MAHKAMAH KONSTITUSI	53
ANGGOTA KABINET/KEMENTERIAN	54
DUTA BESAR	55
GUBERNUR	56
WAKIL GUBERNUR	57
BUPATI	58
WAKIL BUPATI	59
WALIKOTA	60
WAKIL WALIKOTA	61
ANGGOTA DPRD PROVINSI	62
ANGGOTA DPRD KABUPATEN/KOTA	63
DOSEN	64
GURU	65
PILOT	66
PENGACARA	67
NOTARIS	68
ARSITEK	69
AKUNTAN	70
KONSULTAN	71
DOKTER	72
BIDAN	73
PERAWAT	74
APOTEKER	75

PSIKIATER/PSIKOLOG	76
PENYIAR TELEVISI	77
PENYIAR RADIO	78
PELAUT	79
PENELITI	80
SOPIR	81
PIALANG	82
PARANORMAL	83
PEDAGANG	84
PERANGKAT DESA	85
KEPALA DESA	86
BIARAWATI	87
WIRASWASTA	88
LAINNYA	89