# Model Matematika untuk Mendeteksi Penyakit Ginjal Kronis (PGK) dengan metode *Multi Layer Perceptron* (MLP)


Disusun oleh:

| | |
|---|---|
| Almira Farahita Tabina | 1806144336 |
| Anissa Maulidya | 1806144361 |
| Lenni Fitri Anwar | 1806144462 |
| Lista Kurniawati | 1806208011 |
| Muhammad Ulwan Faqih | 1806185903 |
| Nikson Simarmata | 1806144525 |
| Ravelino Nathanael | 1806231102 |
| Sudhanta Dwitama | 1806208314 |
| Syach Riyan Muhammad Ardiyansyah | 1806207854 |
| Syamsyuriani | 1806207942 |
| Tiara Adinda Putri | 1806185891 |

Dibimbing oleh:
Gianinna Ardaneswari, S.Si., M.Si



**UNIVERSITAS INDONESIA**

**DEPOK**

**2021**

**Abstrak**

Penyakit Ginjal Kronis (PGK) menduduki urutan ke-12 penyebab kematian tertinggi di dunia pada tahun 2017. PGK tidak memiliki gejala sehingga banyak penderita yang terlambat mengetahui bahwa ia mengidap PGK. Oleh karena itu, diperlukan konsep *machine learning* untuk mendeteksi PGK dengan lebih cepat. Pada penelitian ini digunakan metode *Multi Layer Perceptron* (MLP) menggunakan data klinis penderita PGK dengan 25 fitur. Setelah dilakukan seleksi fitur didapat 14 fitur yang dapat mendeteksi PGK. Dari data keseluruhan diambil 80% sebagai data *training* yang digunakan untuk merancang model MLP dan sisanya yaitu 20% sebagai data *testing*. Selanjutnya dilakukan *tuning hyperparameter* dengan menggunakan metode *grid search* sehingga diperoleh *number of hidden layer*, *number of (neuron) in hidden layer*, *learning rate* secara berturut-turut 1, 2, 1 sebagai *hyperparameter* terbaik. Terdapat 3 *performance metrics* yang digunakan, yaitu akurasi, presisi, dan *recall*. Rata-rata akurasi sebesar 98,45%, presisi sebesar 98,81%, dan *recall* sebesar 98,52%, dimana ketiganya memberikan nilai yang tinggi dan tidak berbeda jauh, artinya kemampuan model dalam mendeteksi PGK memiliki performa yang seimbang.

**Kata Kunci:** Penyakit ginjal kronis, *Multi Layer Perceptron* (MLP), *machine learning*.

# 1 Pendahuluan

## 1.1 Latar Belakang Masalah

Salah satu organ penting dalam sistem ekskresi pada tubuh manusia adalah ginjal. Sistem ekskresi merupakan sistem dalam tubuh makhluk hidup yang bertugas mengeluarkan zat-zat metabolisme yang sudah tidak diperlukan lagi oleh tubuh [1]. Setiap harinya ginjal memproses sekitar 200 liter darah untuk menyaring atau menghasilkan 2 liter limbah dan sisa cairan yang berlebih dalam bentuk urin. Limbah tersebut dapat berbentuk toksin (racun) [2]. Jika fungsi ginjal terganggu, maka kemampuan untuk menyaring zat-zat sisa dari metabolisme dapat terganggu pula dan akan terjadi penumpukkan sehingga dapat menimbulkan gangguan pada tubuh. Oleh karena itu, ginjal harus dijaga agar tidak terkena penyakit yang berbahaya, salah satunya adalah Penyakit Ginjal Kronis (PGK).

Penyakit Ginjal Kronis (PGK) adalah kerusakan ginjal yang menyebabkan ginjal tidak berfungsi secara optimal untuk membuang racun dan produk sisa di dalam darah yang ditandai dengan eksistensi protein dalam urin dan penurunan laju filtrasi yang terjadi pada glomerulus dalam kurun waktu lebih dari 3 bulan [3]. Pada tahap awal, penderita PGK tidak memiliki gejala. Penderita PGK yang tidak menyadari dirinya menderita PGK dan tidak segera menjalani pengobatan, akan memiliki potensi yang besar untuk mengalami komplikasi atau ESRD (*End Stage Renal Disease*) sehingga dapat menyebabkan kematian [4].

Jumlah penderita PGK di dunia pada tahun 2017 sebanyak 697,5 juta jiwa, sedangkan total kematian akibat PGK pada tahun 2017 sebanyak 1,2 juta jiwa dan menduduki urutan ke-12 penyebab kematian tertinggi di dunia [5]. Oleh karena itu, harus dilakukan pengujian untuk mendeteksi penyakit tersebut. Salah satu parameter untuk menguji PGK adalah perhitungan eGFR (*Estimated Glomerulus Filtration Rate*) [6]. Untuk mendeteksi PGK dalam jumlah data yang besar dibutuhkan waktu yang cukup lama, sehingga digunakan konsep *machine learning* untuk membantu mendeteksi PGK dengan lebih cepat.

Algoritma *machine learning* adalah algoritma yang mampu belajar dari data [7]. *Machine learning* dapat membantu dokter dalam mendeteksi PGK karena output *machine learning* merupakan

model yang bisa memprediksi hasil dari suatu permasalahan, dalam hal ini adalah masalah klasifikasi. Salah satu metode *machine learning* yang dapat menyelesaikan masalah klasifikasi adalah metode *Multi Layer Perceptron* (MLP).

*Multi Layer Perceptron* (MLP) adalah salah satu arsitektur *neural network* yang paling sering digunakan dalam *Medical Decision Support System* (MDSS) [8], karena MLP dapat menyesuaikan berbagai fungsi kontinu dan *non-linear* dengan tingkat akurasi yang sangat tinggi [9].

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, adapun rumusan masalah yang dibahas pada penelitian ini yaitu:

1. Bagaimana mendeteksi PGK dengan metode *Multi Layer Perceptron* (MLP)?
2. Bagaimana performa metode *Multi Layer Perceptron* (MLP) dalam mendeteksi PGK?

## 1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan penelitian ini yaitu:

1. Mendeteksi PGK dengan metode *Multi Layer Perceptron* (MLP).
2. Menganalisis performa metode *Multi Layer Perceptron* (MLP) dalam mendeteksi PGK.

## 1.4 Batasan Masalah

Untuk menyederhanakan permasalahan tanpa mengurangi esensi dari permasalahan pada penelitian, maka ditetapkan batasan masalah sebagai berikut:

1. Data yang digunakan adalah data klinis.
2. Tipe klasifikasi yang digunakan adalah klasifikasi biner, yaitu mengidap PGK atau tidak.
3. Performa metode *Multi Layer Perceptron* (MLP) diukur berdasarkan tingkat akurasi, presisi, dan *recall* model.

# 2 Model Matematika yang Digunakan

## 2.1 Pengenalan *Machine Learning*

Algoritma *machine learning* adalah algoritma yang mampu belajar dari data [7]. Ada 3 jenis tipe data yang digunakan pada machine learning yaitu data *training*, data *testing*, dan data *validation*. Data *training* adalah data yang digunakan untuk *training* model. Data *testing* adalah data independen yang tidak terlibat dalam pembentukan model. Data *validation* digunakan untuk proses validasi model dan mencegah *overfitting*. *Output* dari *machine learning* adalah sebuah model yang bisa memprediksi hasil dari suatu permasalahan [10]. Berdasarkan tipe pembelajarannya terdapat 3 jenis model *machine learning* yaitu[7]:

a. ***Supervised Learning***
*Supervised learning* merupakan jenis *machine learning* yang membangun suatu model *machine learning*, dengan data *training* disertai label sebagai target pembelajaran.

b. ***Unsupervised Learning***
*Unsupervised learning* merupakan jenis *machine learning* yang membangun suatu model *machine learning*, dengan data *training* tidak disertai target atau label.

c. ***Reinforcement Learning***
*Reinforcement learning* merupakan jenis *machine learning* yang membangun suatu model *machine learning* yang dapat membuat suatu urutan *action* untuk memaksimumkan *reward*.

Pada *machine learning* juga terdapat 2 jenis klasifikasi yaitu[11]:

a. ***Multiclass Classification***
*Multiclass classification* mengacu pada jenis klasifikasi yang memiliki lebih dari dua label kelas. *Multiclass classification* biasanya menghasilkan banyak kelas tergantung dengan masalah yang ingin diselesaikan.

b. ***Binary classification***
*Binary classification* mengacu pada jenis klasifikasi yang memiliki dua label kelas. Kelas untuk normal diberi label kelas 0 dan kelas yang tidak normal diberi label kelas 1.

## 2.2 Pengenalan *Multi Layer Perceptron* (MLP)

*Multi Layer Perceptron* (MLP) adalah salah satu arsitektur (*neural network*) yang paling sering digunakan dalam *Medical Decision Support System* (MDSS), dan termasuk tipe *supervised neural networks* [8]. MLP merupakan *Feedforward Neural Network* (FNN) yang mana terdiri dari dua tahap, yaitu *feedforward* dan *backpropagation*. Pada tahap *feedforward*, sinyal *input* diberikan ke dalam jaringan. Komputasi dilakukan oleh *neuron* pada setiap *layer* sehingga menghasilkan *output*. Pada tahap ini, bobot pada jaringan tidak mengalami pengubahan. Sebaliknya, pada tahap *backpropagation*, semua bobot diperbaiki berdasarkan *error* jaringan. Bobot diperbaiki untuk membuat *output* yang dihasilkan semakin mendekati *output* yang diinginkan [7].

Seperti halnya *neural network*, MLP juga terdiri dari satu *(*input layer), satu atau lebih dari satu *hidden layer* dengan fungsi aktivasi *non-linear*, dan satu *output layer* [8]. Berikut penjelasan masing-masing layer [12]:

1. *Input layer*
   *Input layer* merupakan *layer* untuk suatu data akan diinput ke dalam sistem (inisialisasi *input*).

2. *Hidden layer*
   *Hidden layer* merupakan *layer* yang terletak di antara *input layer* dan *output layer*. *Layer* ini membantu proses pengolahan data. Semakin banyak *hidden layer* digunakan, waktu yang dibutuhkan untuk memperoleh *output* semakin sedikit.

3. *Output layer*
   *Output layer* merupakan *layer* yang terdapat hasil dari proses suatu sistem, seperti yang menghasilkan suatu klasifikasi.

4. Fungsi aktivasi
   Fungsi aktivasi mempunyai beberapa karakteristik yang harus dipenuhi, yaitu tidak turun, kontinu, dan diturunkan. Pada *hidden layer*, fungsi aktivasi *non-linear* yang sering digunakan adalah fungsi ReLu atau fungsi sigmoid. Pada *output layer*, untuk masalah regresi, fungsi aktivasi yang biasa digunakan adalah fungsi identitas. Untuk masalah klasifikasi, fungsi aktivasi yang biasa digunakan adalah fungsi sigmoid atau fungsi softmax.

Berikut beberapa fungsi aktivasi yang sering digunakan [7]:

- Fungsi ReLu (*Rectified Linear Unit*)

$$r(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{1}$$

- Fungsi Sigmoid

$$\sigma(a) = \frac{1}{1 + e^{1/a}} \tag{2}$$

- Fungsi Hyperbolic Tangent

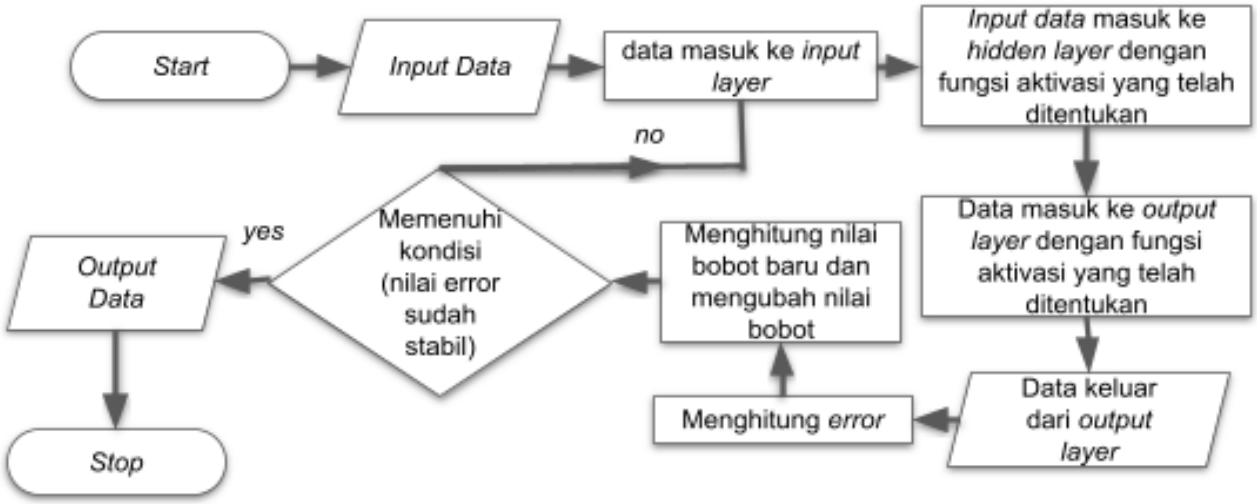
$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{3}$$

- Fungsi Softmax

$$l(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{(z_i)}} \tag{4}$$

## 2.3 Cara kerja *Multi Layer Perceptron* (MLP)

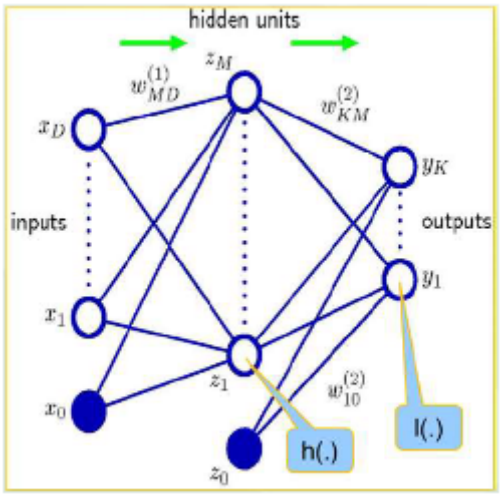
Cara kerja *Multi Layer Perceptron* (MLP) memiliki jalur *flowchart* sebagai berikut




Gambar 1: *Flowchart* cara kerja MLP

Terdapat 3 tahap utama MLP yaitu dari tahap *input* data sampai dengan data keluar dari *output layer* merupakan tahap *feedforward* dan dari tahap menghitung *error*, hingga memperbarui nilai bobot merupakan tahap *backpropagation*.

### 2.3.1 *Feedforward Propagation*




Gambar 2: Arsitektur MLP

Diberikan *input* data $x = (x_1, x_2, \ldots, x_D)^T$ dengan $D$ fitur, satu *hidden layer* terdiri dari $M$ *neuron* dengan fungsi aktivasi $h(.)$, serta $k$ *output* target $y_1, y_2, \ldots, y_K$ dengan fungsi aktivasi $l(.)$ [7].

Pada penelitian ini digunakan fungsi ReLu sebagai fungsi aktivasi pada *hidden layer* dan fungsi sigmoid sebagai fungsi aktivasi pada *output layer*. Fungsi aktivasi sigmoid merupakan salah satu fungsi yang digunakan untuk masalah klasifikasi biner, seperti pada kasus deteksi PGK. Dari beberapa jurnal penelitian yang membahas tentang PGK, fungsi aktivasi sigmoid memberikan tingkat akurasi yang tinggi, yaitu 98,75%.

Aliran data masuk ke model MLP sebagai berikut:

- Data masuk ke *neuron* $z_j$:

$$a_j = \sum_{i=1}^{D} w_{ij}^{(1)} x_i \tag{5}$$

- Data keluar dari *neuron* $z_j$:

$$z_j = \begin{cases} a_j, & a_j > 0 \\ 0, & a_j \leq 0 \end{cases} \tag{6}$$

- Data masuk ke *neuron* $y_k$:

$$b_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j \tag{7}$$

- Data keluar dari *neuron* $y_k$:

$$y_k = \sigma(b_k) = \frac{1}{1 + e^{1/b_k}} \tag{8}$$

Secara matematis, aliran data pada MLP di atas memiliki bentuk sebagai berikut:

$$y_k(x) = \begin{cases} \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} \sum_{i=1}^{D} w_{ij}^{(1)} x_i), & a_j > 0 \\ 0, & a_j \leq 0 \end{cases} \tag{9}$$

dimana

$y_k$ adalah *output* target ke $k$
$w_{ij}^{(1)}, w_{kj}^{(2)}$ adalah parameter bobot
$\sigma(x)$ adalah fungsi aktivasi yang digunakan, yaitu fungsi sigmoid.

Sehingga, secara matematis MLP adalah suatu model *non-linear* dari parameter bobot [7].

### 2.3.2 Fungsi *error*

Jumlah *error* dari semua data *training* dapat dinyatakan dengan menggunakan *sum of square error* (SSE) berikut:

$$E(\overline{w}) = \sum_{n=1}^{N} E_n(\overline{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_k - t_k)^2 \tag{10}$$

Sehingga masalah pembelajaran pada MLP adalah bagaimana menentukan parameter bobot $\overline{w}$ sedemikian sehingga $E(\overline{w})$ adalah minimum.

Dengan menurunkan fungsi *error* diperoleh hasil berikut:

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = (y_k - t_k) z_j^2 (1 - z_j) \tag{11}$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \begin{cases} \sum_{k=1}^{K} (y_k - t_k) z_j (1 - z_j) w_{kj}^{(2)} x_i, & a_j > 0 \\ 0, & a_j <= 0 \end{cases} \tag{12}$$

Misalkan

$$\delta_k = y_k - t_k \tag{13}$$

dan

$$\delta_j = \begin{cases} z_j(1 - z_j) \sum_{k=1}^{K} w_{kj}^{(2)} \delta_k, & a_j > 0 \\ 0, & a_j <= 0 \end{cases} \tag{14}$$

Maka

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j^2 (1 - z_j) \quad \text{dan} \quad \frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

### 2.3.3 *Backpropagation*

*Error* pada setiap *(*neuron) $k$, $k = 1, 2, ..., K$ pada *output layer* diberikan oleh formula:

$$\delta_k = y_k - t_k \tag{15}$$

Sehingga pembaruan nilai bobot pada *output layer* adalah sebagai berikut:

$$w_{kj}^{(2)(\tau+1)} = w_{kj}^{(2)(\tau)} - \eta \delta_k z_j^2 (1 - z_j) \tag{16}$$

dengan

$\eta$ adalah *learning rate*
$\tau$ adalah banyaknya pembaruan yang telah dilakukan

*Error* setiap *(*neuron) $j$, $j = 1, 2, ..., M$ pada *hidden layer* diberikan oleh formula:

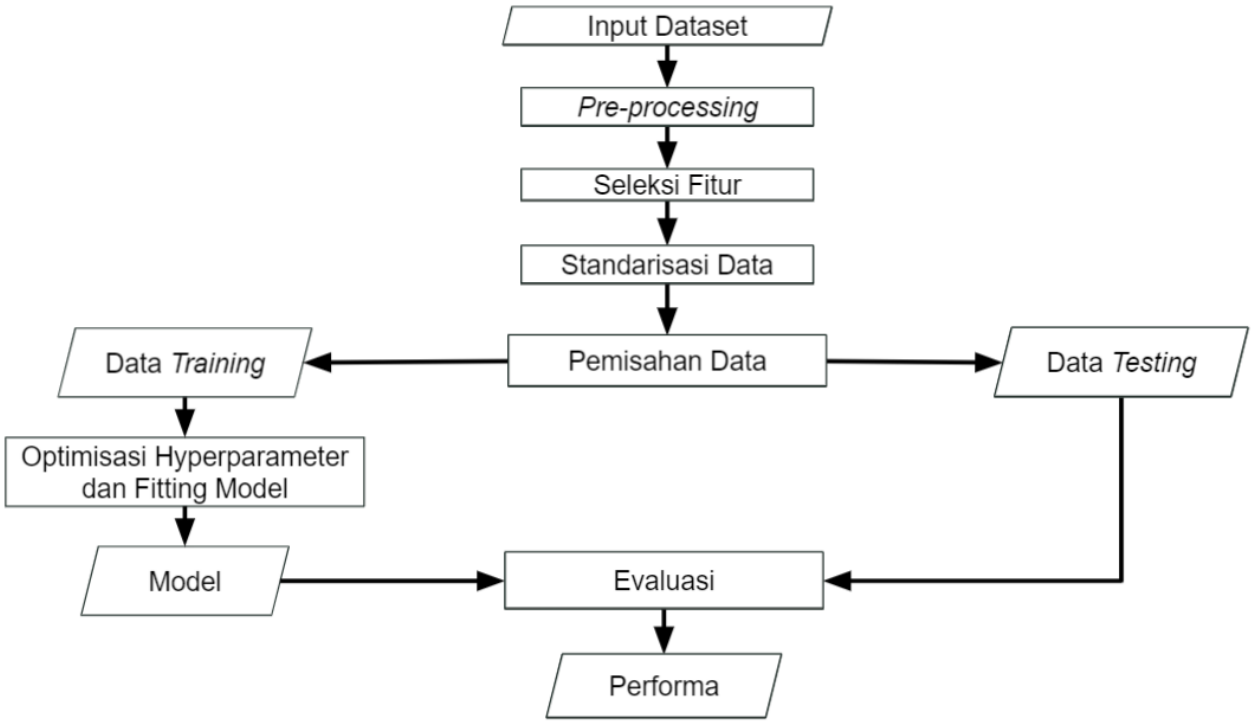
$$\delta_j = \begin{cases} z_j(1 - z_j) \sum_{k=1}^{K} w_{kj}^{(2)} \delta_k, & a_j > 0 \\ 0, & a_j <= 0 \end{cases} \tag{17}$$

Sehingga pembaruan nilai bobot pada *hidden layer* adalah sebagai berikut:

$$w_{ji}^{(1)(\tau+1)} = w_{ji}^{(1)(\tau)} - \eta \delta_j x_i \tag{18}$$

# 3 Rancangan dan Algoritma Penelitian

## 3.1 Tahapan Umum Simulasi




Gambar 3: Tahapan Umum Simulasi Model MLP

## 3.2 Data

*Dataset* yang digunakan pada penelitian ini diperoleh dari UCI *Repository Machine Learning* [22]. *Dataset* tersebut berupa data klinis 400 data responden dari Apollo Hospital yang terdiri dari 250 data yang terkena PGK dan 150 data yang tidak terkena PGK. *Dataset* terdiri dari 24 fitur dan 1 kelas target (*ckd/notckd*). Berikut ini ditunjukkan contoh data klinis responden yang digunakan.

| id | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | bu | sc | sod | pot | hemo | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | 121.0 | 36.0 | 1.2 | NaN | NaN | 15.4 | 44.0 | 7800.0 | 5.2 | yes | yes | no | good | no | no | ckd |
| 1 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | NaN | 18.0 | 0.8 | NaN | NaN | 11.3 | 38.0 | 6000.0 | NaN | no | no | no | good | no | no | ckd |
| 2 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | 423.0 | 53.0 | 1.8 | NaN | NaN | 9.6 | 31.0 | 7500.0 | NaN | no | yes | no | poor | no | yes | ckd |
| 3 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | 117.0 | 56.0 | 3.8 | 111.0 | 2.5 | 11.2 | 32.0 | 6700.0 | 3.9 | yes | no | no | poor | yes | yes | ckd |
| 4 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 106.0 | 26.0 | 1.4 | NaN | NaN | 11.6 | 35.0 | 7300.0 | 4.6 | no | no | no | good | no | no | ckd |

| id | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | bu | sc | sod | pot | hemo | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | 121.0 | 36.0 | 1.2 | NaN | NaN | 15.4 | 44.0 | 7800.0 | 5.2 | yes | yes | no | good | no | no | ckd |
| 1 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | NaN | 18.0 | 0.8 | NaN | NaN | 11.3 | 38.0 | 6000.0 | NaN | no | no | no | good | no | no | ckd |
| 2 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | 423.0 | 53.0 | 1.8 | NaN | NaN | 9.6 | 31.0 | 7500.0 | NaN | no | yes | no | poor | no | yes | ckd |
| 3 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | 117.0 | 56.0 | 3.8 | 111.0 | 2.5 | 11.2 | 32.0 | 6700.0 | 3.9 | yes | no | no | poor | yes | yes | ckd |
| 4 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 106.0 | 26.0 | 1.4 | NaN | NaN | 11.6 | 35.0 | 7300.0 | 4.6 | no | no | no | good | no | no | ckd |

Gambar 4: Contoh data klinis responden yang digunakan

## 3.3 *Pre-processing*

*Pre-processing* dilakukan untuk menangani *error* dan *missing value* pada *dataset*. Jenis *missing value* pada *dataset* PGK adalah *Missing at Random* (hilang secara acak). Oleh karena itu, dapat dilakukan beberapa cara untuk menanganinya [8]:

1. Memperkirakan Nilai dari Fitur Lain
   Cara ini biasanya digunakan untuk memperkirakan data kualitatif dari data kuantitatif yang saling berhubungan atau sebaliknya. Pada *dataset* PGK, terdapat beberapa fitur yang saling berhubungan dan dapat merepresentasikan satu sama lain. Sehingga, nilai *missing value* dari suatu fitur dapat diperkirakan dari fitur yang berhubungan dengan fitur tersebut.

2. Mean atau Median atau Modus
   Mean, median, dan modus digunakan untuk memperkirakan *missing value* pada fitur dengan melihat distribusi data pada fitur tersebut. Semakin bagus distribusi dari data, maka akan semakin akurat nilai yang diperkirakan. Mean akan terlebih dahulu diprioritaskan bila penyebaran datanya sudah cukup bagus dan tidak terdapat *outlier*, jika terdapat *outlier* maka mean akan bergeser sehingga akurasinya akan berkurang. Untuk mengatasi masalah tersebut, median dan modus dapat menjadi pilihan sebagai alternatif dari mean. Cara ini bisa menjadi pilihan untuk data kuantitatif dan tidak dapat direpresentasikan oleh fitur lainnya.

3. Regresi Linear
   Regresi linear adalah salah satu cara imputasi yang paling akurat. Cara ini dilakukan untuk dua fitur kuantitatif yang memiliki korelasi yang cukup tinggi. *Missing value* pada kedua fitur yang memiliki korelasi tinggi kemudian diimputasi berdasarkan kecenderungan umum dari model linear sederhana yang terbentuk dari kedua fitur.

*Dataset* PGK berjumlah 400 data yang terdiri dari 24 fitur dan 1 kelas target dengan cara penanganan tiap fitur sebagai berikut:

| **No** | **Regresi Linear** | **Mean/ Median/ Modus** | **Perkiraan dari Nilai Fitur Lain** | **Menghapus Baris** | **Menghapus Kolom** |
|---|---|---|---|---|---|
| 1 | *Packed Cell Volume* | *Age* | *Blood Pressure* | *Albumin* | *Red Blood Cells* |
| 2 | *Red Blood Cell Count* | *Specific Gravity* | *Pus Cell* | *Sugar* | - |
| 3 | - | *Blood Urea* | *Pus Cell Clumps* | - | - |
| 4 | - | *Serum Creatinine* | *Bacteria* | - | - |
| 5 | - | *Sodium* | *Blood Glucose Random* | - | - |
| 6 | - | *Potassium* | *Hypertension* | - | - |
| 7 | - | *Hemoglobin* | *Diabetes Mellitus* | - | - |

| | | | | | |
|---|---|---|---|---|---|
| 8 | - | *White Blood Cell Count* | *Coronary Artery Disease* | - | - |
| 9 | - | - | *Appetite* | - | - |
| 10 | - | - | *Pedal Edema* | - | - |
| 11 | - | - | *Anemia* | - | - |

Tabel 1: Cara penanganan untuk masing-masing fitur

Sehingga *dataset* setelah *preprocessing* berjumlah 351 data yang terdiri dari 23 fitur dan 1 kelas target.

## 3.4 Seleksi Fitur

Salah satu hal yang perlu diperhatikan dalam pembuatan model yaitu seleksi fitur. Seleksi fitur adalah proses memilih fitur-fitur yang paling berpengaruh pada variabel prediksi atau output yang diinginkan [13].

Seleksi fitur diperlukan karena [14] :

1. Proses training data pada machine learning lebih cepat.
2. Mengurangi kompleksitas model dan membuat model lebih mudah diinterpretasikan.
3. Meningkatkan akurasi dari model jika variabel-variabel yang dipilih tepat.
4. Mengurangi overfitting atau variabel yang tidak signifikan pada PGK.

Pada penelitian ini, metode seleksi fitur yang digunakan adalah metode *backward selection*. Metode *backward selection* memiliki *running* lebih cepat dibandingkan dengan *best selection*. Metode *backward selection* adalah metode yang dimulai dengan menggunakan model yang memuat semua variabel, selanjutnya variabel yang kurang signifikan akan dihapus dari model berdasarkan kriteria $C_p$ [15]. Kriteria $C_p$ didefinisikan sebagai berikut:

$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2) \tag{19}$$

$$RSS = \sum_{k=1}^{n}(\hat{y_k} - y_k)^2 \tag{20}$$

dengan

$d$ adalah jumlah fitur
$\hat{\sigma}^2$ adalah perkiraan variansi dari *error*
$\hat{y_k}$ adalah hasil prediksi model
$y_k$ adalah target yaitu klasifikasi PGK
$n$ adalah banyaknya sampel.

Setelah diterapkan pada *dataset*, diperoleh nilai $C_p$ dari 23 model yang diperoleh berdasarkan banyak fiturnya sebagai berikut:

| | Cp | Fitur |
|---|---|---|
| 1 | 0.246226 | [sg] |
| 2 | 0.116292 | [sg, hemo] |
| 3 | 0.101896 | [sg, hemo, dm] |
| 4 | 0.093457 | [sg, al, hemo, dm] |
| 5 | 0.088823 | [sg, al, hemo, wc, dm] |
| 6 | 0.087138 | [bp, sg, al, hemo, wc, dm] |
| 7 | 0.085475 | [bp, sg, al, bu, hemo, wc, dm] |
| 8 | 0.084205 | [bp, sg, al, bu, hemo, wc, rc, dm] |
| 9 | 0.083170 | [bp, sg, al, bgr, bu, hemo, wc, rc, dm] |
| 10 | 0.082546 | [bp, sg, al, bgr, bu, sc, hemo, wc, rc, dm] |
| 11 | 0.082226 | [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, dm] |
| 12 | 0.082027 | [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, dm, ane] |
| 13 | 0.081883 | [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, dm, cad, ane] |
| 14 | 0.081785 | [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, htn, dm, cad, ane] |
| 15 | 0.081901 | [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, htn, dm, cad, appet, ane] |
| 16 | 0.082154 | [bp, sg, al, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, ane] |
| 17 | 0.082525 | [bp, sg, al, su, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, ane] |
| 18 | 0.082904 | [bp, sg, al, su, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |
| 19 | 0.083315 | [bp, sg, al, su, ba, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |
| 20 | 0.083721 | [bp, sg, al, su, pcc, ba, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |
| 21 | 0.084144 | [age, bp, sg, al, su, pcc, ba, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |
| 22 | 0.084571 | [age, bp, sg, al, su, pc, pcc, ba, bgr, bu, sc, sod, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |
| 23 | 0.084571 | [age, bp, sg, al, su, pc, pcc, ba, bgr, bu, sc, sod, pot, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane] |

Gambar 5: Daftar 23 model terbaik

Dari 23 model tersebut diperoleh nilai $C_p$ terkecil adalah model dengan 14 fitur. Fitur yang digunakan yaitu sebagai berikut:

```
Cp                                                          0.0817853
Fitur    [bp, sg, al, bgr, bu, sc, hemo, pcv, wc, rc, htn, dm, cad, ane]
Name: 14, dtype: object
```

Gambar 6: Model terbaik dengan $C_p$ terkecil

Sehingga dapat dijabarkan data klinis yang telah melalui seleksi fitur

| | bp | sg | al | bgr | bu | sc | hemo | pcv | wc | rc | htn | dm | cad | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 80 | 1.020 | 1 | 121 | 36.0 | 1.2 | 15.4 | 44.0 | 7800 | 5.200000 | 1 | 1 | 0 | 0 | 1 |
| **1** | 50 | 1.020 | 4 | 122 | 18.0 | 0.8 | 11.3 | 38.0 | 6000 | 4.232106 | 0 | 0 | 0 | 0 | 1 |
| **2** | 80 | 1.010 | 2 | 423 | 53.0 | 1.8 | 9.6 | 31.0 | 7500 | 3.773686 | 0 | 1 | 0 | 1 | 1 |
| **3** | 70 | 1.005 | 4 | 117 | 56.0 | 3.8 | 11.2 | 32.0 | 6700 | 3.900000 | 1 | 0 | 0 | 1 | 1 |
| **4** | 80 | 1.010 | 2 | 106 | 26.0 | 1.4 | 11.6 | 35.0 | 7300 | 4.600000 | 0 | 0 | 0 | 0 | 1 |

Gambar 7: Contoh data setelah melalui proses seleksi fitur

Skala yang berbeda dari fitur data mempengaruhi pemodelan, yang dapat berakibat buruk pada hasil prediksi yang bias dalam hal *misclassification* dan tingkat akurasi. Oleh karena itu, perlu

dilakukan penskalaan data sebelum pemodelan. Oleh karena itu dilakukan standarisasi data agar model dapat belajar dengan lebih cepat dan meningkatkan akurasi model. Standarisasi data dilakukan dengan *StandardScaler* agar data berdistribusi normal standar, yaitu memiliki mean 0 dan dan standar deviasi 1.

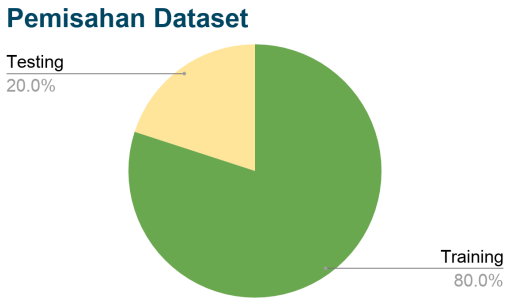
| | bp | sg | bgr | bu | sc | hemo | pcv | wc | rc | al | htn | dm | cad | ane |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.301532 | 0.448300 | -0.341271 | -0.378493 | -0.385674 | 1.000587 | 0.606227 | 0.266754 | 0.629524 | 1 | 1 | 1 | 0 | 0 |
| **1** | -1.966422 | 0.448300 | -0.328218 | -0.752422 | -0.498390 | -0.551412 | -0.126860 | -0.145825 | -0.450583 | 4 | 0 | 0 | 0 | 0 |
| **2** | 0.301532 | -1.307875 | 3.600691 | -0.025337 | -0.216601 | -1.194924 | -0.982128 | 0.197991 | -0.962149 | 2 | 0 | 1 | 0 | 1 |
| **3** | -0.454452 | -2.185962 | -0.393482 | 0.036985 | 0.346979 | -0.589266 | -0.859947 | 0.014622 | -0.821191 | 4 | 1 | 0 | 0 | 1 |
| **4** | 0.301532 | -1.307875 | -0.537064 | -0.586231 | -0.329317 | -0.437851 | -0.493404 | 0.152149 | -0.040037 | 2 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **346** | 0.301532 | 0.448300 | -0.093267 | -0.108432 | -0.582927 | 1.114148 | 0.972770 | 0.014622 | 0.294744 | 0 | 0 | 0 | 0 | 0 |
| **347** | -0.454452 | 1.326387 | -0.941702 | -0.482362 | -0.385674 | 1.416977 | 1.828038 | 0.266754 | 1.745459 | 0 | 0 | 0 | 0 | 0 |
| **348** | 0.301532 | 0.448300 | -0.615381 | -0.586231 | -0.554748 | 1.152002 | 1.217132 | -0.008299 | 0.852711 | 0 | 0 | 0 | 0 | 0 |
| **349** | -1.210437 | 1.326387 | -0.432641 | -0.087659 | -0.442032 | 0.546343 | 1.461495 | 0.129228 | 1.410678 | 0 | 0 | 0 | 0 | 0 |
| **350** | 0.301532 | 1.326387 | -0.210742 | -0.752422 | -0.413853 | 1.152002 | 1.705857 | 0.037544 | 1.633865 | 0 | 0 | 0 | 0 | 0 |

Gambar 8: Contoh data setelah melalui proses standarisasi data

## 3.5 Pemisahan Data

Pemisahan *dataset* PGK dilakukan dengan memisahkan *dataset* menjadi data *training* dan data *testing*. Puspitasari dkk, melakukan penelitian terkait klasifikasi penyakit gigi dan mulut menggunakan metode SVM dimana perbandingan rasio data dengan tingkat akurasi terbaik terdapat pada rasio data *training:testing* 80%:20% dengan nilai akurasi sebesar 93,328% [16]. Abushariah dkk., melakukan penelitian terkait sistem diagnosis penyakit jantung otomatis berbasis pendekatan *Artificial Neural Network* (ANN) dan *Adaptive Neuro-Fuzzy Inference Systems* (ANFIS) dengan perbandingan rasio data *training:testing* 80%:20% diperoleh nilai akurasi sebesar 87,04% [17]. Lin dkk, melakukan penelitian terkait diagnosis penyakit hati dengan CNN dan NN dengan perbandingan rasio data *training:testing* 80%:20% diperoleh nilai akurasi sebesar 78,65% dan 86,40% (setelah *balancing* data) [18].

Berdasarkan ketiga penelitian yang pernah dilakukan oleh peneliti sebelumnya, didapatkan rasio data *training:testing* berupa 80%:20% memberikan tingkat akurasi yang cukup baik. Oleh karena itu, pada penelitian ini *dataset* akan dipisah menjadi 80% data *training* dan 20% data *testing*.




Gambar 9: Pemisahan *Dataset*

*Dataset* yang digunakan berisi 351 data responden, dengan sebanyak 206 responden PGK dan sebanyak 145 responden normal. Berdasarkan pemisahan *dataset* yang telah ditentukan seperti Gambar (9), 80% dari *dataset* akan digunakan sebagai data *training*, yang artinya akan ada sebanyak 280 data responden yang terbagi atas 165 data responden PGK dan 115 data responden normal. Sementara sisa 20% dari *dataset* akan digunakan untuk data *testing*, yang artinya akan ada sebanyak 71 data responden yang terbagi atas 41 data responden PGK dan 30 data responden normal.

| **351 data** | | | |
|---|---|---|---|
| **Data Training** | | **Data Testing** | |
| **280 data** | | **71 data** | |
| 165 data PGK | 115 data normal | 41 data PGK | 30 data normal |

Tabel 2: Pemisahan *dataset*

## 3.6 Parameter dan *Hyperparameter Tuning*

### 3.6.1 *Cross Validation*

*Cross validation* yang dikenal juga sebagai *k-fold* CV adalah salah satu proses validasi model untuk mengestimasi performa dari sebuah model. Proses ini dimulai dengan mempartisi data menjadi $k$ grup atau *folds* dengan ukuran sama. Untuk setiap *fold* ke-$k$ proses *fitting* model akan dilakukan terhadap *k-1 fold* lainnya, kemudian akan dihitung akurasi (menggunakan fungsi *loss* pada persamaan (10)) menggunakan *fold* ke-$k$ tersebut. Maka akan diperoleh $k$ jumlah akurasi $a_1, a_2, \ldots, a_k$. Estimasi yang akan digunakan untuk model tersebut adalah [15]:

$$Akurasi\, CV = \frac{1}{k} \sum_{i=1}^{k} a_i \tag{21}$$

### 3.6.2 Penentuan *Hyperparameter* untuk Optimisasi

*Hyperparameter* yang akan dioptimasi adalah sebagai berikut:

| **Parameter-parameter pada model** | |
|---|---|
| ***Number of Hidden Layer*** | Banyaknya *layer* yang digunakan pada *hidden layer*. *Number of Hidden Layer* merupakan salah satu parameter yang akan dioptimasi. |
| ***Number of neuron in Hidden Layer*** | Banyaknya *neuron* yang digunakan pada *hidden layer*. *Number of neuron in Hidden Layer* merupakan salah satu parameter yang akan dioptimasi. |
| ***Activation Function*** | Fungsi aktivasi yang digunakan pada *hidden layer* adalah fungsi ReLU, sedangkan fungsi aktivasi yang digunakan pada *output layer* adalah fungsi *sigmoid*. |
| ***Learning Rate*** | Digunakan untuk mempercepat laju pembelajaran dalam menentukan bobot yang optimal. *Learning rate* merupakan salah satu parameter yang akan dioptimasi. |

| | |
|---|---|
| ***Epoch*** | Jumlah pengulangan proses pembelajaran dalam satu *dataset*. |
| ***Batch Size*** | Jumlah sampel data yang diproses pada satu *epoch*. |

Tabel 3: Parameter pada model MLP

Calon *hyperparameter* yang akan dioptimasi adalah sebagai berikut:

| **Parameter yang akan dioptimasi pada model MLP** | | | **Rujukan** |
|---|---|---|---|
| ***Layer*** | **Parameter** | **Nilai** | |
| ***Hidden Layer*** | *Number of Hidden Layer* | [1,2,3,4,5] | [7] |
| | *Number of neuron in Hidden Layer* | [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] | [7] |
| ***Optimasi SGD*** | *Learning Rate* | [0,0001, 0,001, 0,01, 0,1, 1] | [19] |

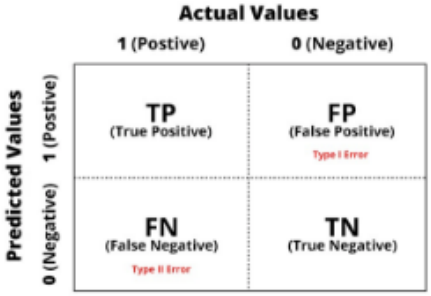
Tabel 4: *Hyperparameter* yang akan dioptimasi

Selanjutnya, dicari kombinasi *hyperparameter* terbaik dari calon-calon *hyperparameter* dengan menggunakan *GridSearchCV* dengan *cross validation* sebanyak 10-*fold*. Metode *Grid Search* akan mencari seluruh kemungkinan kombinasi dari kandidat parameter yang ada, yaitu 350 kombinasi untuk digunakan dalam pembentukan model. Dari total 350 model yang telah dibentuk tersebut akan dipilih model yang memiliki nilai akurasi terbaik. Diperoleh kombinasi nilai *hyperparameter* terbaik yang akan digunakan adalah *learning rate* = 1, *Number of Hidden Layer* = 1, dan *Number of neuron in Hidden Layer* = 2.

# 4 Pembahasan

## 4.1 Performa Model MLP

Performa diperlukan untuk mengetahui seberapa baik suatu model, akan dilakukan peninjauan parameter pengukuran performa dengan menggunakan *confusion matrix*. *Confusion matrix* atau *error matrix* merupakan suatu formula dalam bentuk tabel untuk membandingkan hasil klasifikasi yang dilakukan oleh model dengan hasil yang sebenarnya.[20]




Gambar 10: *Confusion Matrix*

Dari Gambar 10 dapat dilihat terdapat 4 istilah sebagai representasi hasil klasifikasi di model yang kita punya pada *confusion matrix*. Keempat istilah tersebut adalah[21] :

- ***True Positive* (TP)**
  Merupakan data positif yang diprediksi benar.
- ***True Negative* (TN)**
  Merupakan data negatif yang diprediksi benar.
- ***False Positive* (FP) — *Type I Error***
  Merupakan data negatif namun diprediksi sebagai data positif.
- ***False Negative* (FN) — *Type II Error***
  Merupakan data positif namun diprediksi sebagai data negatif.

Dari tabel *confusion matrix* diatas, dapat digunakan untuk membandingkan hasil klasifikasi yang dilakukan oleh model dengan hasil yang sebenarnya, dengan cara menghitung *performance metrics*. *Performance metrics* yang sering digunakan adalah *accuracy*, presisi dan *recall*.

- ***Accuracy***
  Akurasi menginterpretasikan seberapa akurat model dapat mengklasifikasikan dengan benar. Akurasi merupakan rasio antara prediksi benar (positif dan negatif) dan keseluruhan data.
- **Presisi**
  Presisi menginterpretasikan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Presisi merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.
- ***Recall***
  Recall menginterpretasikan keberhasilan model dalam menemukan kembali sebuah informasi. Recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif digabungkan dengan data yang salah negatif.

Nilai akurasi, presisi, dan *recall* dapat diperoleh dengan persamaan sebagai berikut:

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

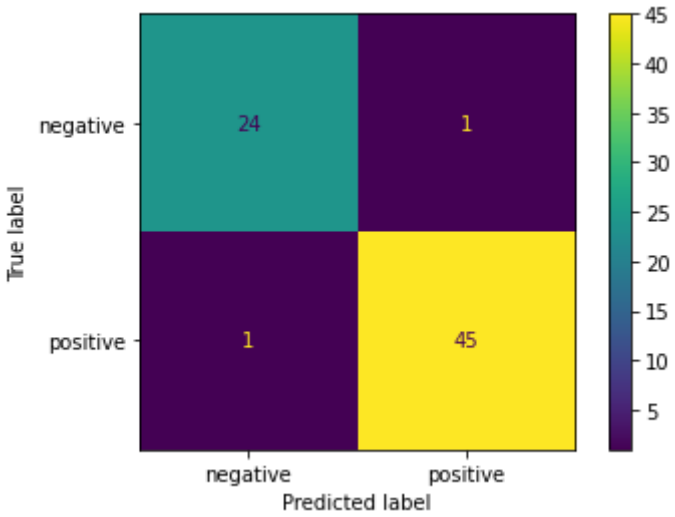
$$Recall = \frac{TP}{TP + FN}$$

dimana:

TP : banyaknya data yang *True Positive*
TN : banyaknya data yang *True Negative*
FP : banyaknya data yang *False Positive*
FN : banyaknya data yang *False Negative*

## 4.2 Analisis Performa Model

Untuk memperoleh nilai TP, TN, FP, dan FN digunakan bantuan aplikasi *Google Colaboratory Research* sehingga diperoleh *confusion matrix* sebagai berikut :




Gambar 11: Visualisasi *confusion matrix*

| ***Predcited Value*** / ***Actual Value*** | **0 (Negatif)** | **1 (Positif)** |
|---|---|---|
| **0 (Negatif)** | True Negative = 24 | False Positive = 1 |
| **1 (Positif)** | False Negative = 1 | True Positive = 46 |

Tabel 5: Hasil *Confusion Matrix*

Pada tabel *confusion matrix*, diperoleh 4 nilai berbeda, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). TP berjumlah 46 adalah banyak data yang diprediksi mengidap PGK dan data sesungguhnya juga mengidap PGK. TN berjumlah 24 adalah banyak data yang diprediksi tidak mengidap PGK dan data sesungguhnya juga tidak mengidap PGK. FP berjumlah 1 adalah banyak data yang diprediksi mengidap PGK tapi data sesungguhnya tidak mengidap PGK. FN berjumlah 1 adalah banyak data yang diprediksi tidak mengidap PGK tapi data sesungguhnya mengidap PGK. Kesalahan tipe I merupakan banyak FP sedangkan kesalahan tipe II merupakan banyak FN. Berdasarkan tabel diatas, model yang dikonstruksi memiliki banyak kesalahan tipe I dan kesalahan tipe II yang cukup seimbang.

```
Akurasi: 0.971830985915493
Presisi: 0.9782608695652174
Recall: 0.9782608695652174
```

Gambar 12: Hasil akurasi, presisi dan *recall* model

Dari implementasi program yang dijalankan diperoleh hasil akurasi, presisi, dan *recall* dari model yang digunakan diperoleh berturut-turut adalah 97,18% ; 97,78% ; 97,82% . Nilai presisi dan *recall* tidak berbeda jauh dengan nilai akurasi yang diperoleh. Sehingga kemampuan model MLP dalam klasifikasi memiliki performa yang seimbang.

Dilakukan 10 kali proses pembelajaran yang menggunakan data *training* dan data *testing* yang berbeda agar mencerminkan konsistensi kinerja dari masing-masing model. Berikut adalah rata-rata hasil model MLP untuk 10 kali proses pembelajaran:

```
Rata-rata akurasi untuk 10 kali pembelajaran: 0.9845070422535211
Rata-rata presisi untuk 10 kali pembelajaran: 0.9881294944391446
Rata-rata recall untuk 10 kali pembelajaran: 0.9852268142058274
```

Gambar 13: Rata-rata akurasi, presisi dan *recall*

## 4.3 Analisis *Overfit* Model

Suatu model akan dikatakan overfit jika akurasi yang diperoleh data *training* tinggi akan tetapi akurasi pada data *validation* rendah.

| **Data yang Digunakan** | **Akurasi** |
|---|---|
| Data *training* | 0.99603175 |
| Data *validation* | 0.98928571 |

Tabel 6: Akurasi data *training* dan data *validation*

Berdasarkan perhitungan yang dilakukan, didapatkan tingkat akurasi data *training* sebesar 99,6% dan tingkat akurasi data *validation* sebesar 98,9%. Diperoleh perbedaan skor sekitar 0,675%. Karena perbedaan skor akurasi yang cukup kecil inilah yang menandakan penggunaan model pada data *validation* tidak menunjukkan performa yang buruk. Sedemikian sehingga diperoleh bahwa model tidak *overfit*.

# 5 Penutup

## 5.1 Kesimpulan

1. Pada penelitian ini telah dibentuk pemodelan untuk deteksi dini penyakit ginjal kronis dengan metode MLP. Model yang diperoloh adalah model MLP dengan satu *input layer* yang terdiri dari 14 *neuron*, satu *hidden layer* yang terdiri dari 2 *neuron* dengan fungsi aktivasi ReLu, dan satu *output layer* yang terdiri dari 1 *neuron* dengan fungsi aktivasi sigmoid.

2. Berdasarkan hasil simulasi yang telah dilakukan disimpulkan bahwa kinerja model MLP menghasilkan rata-rata akurasi model sebesar 98,45%, presisi 98,81% dan *recall* 98,52%, dengan ketiganya memberikan nilai yang tinggi dan tidak berbeda jauh, artinya kemampuan model dalam mendeteksi PGK memiliki performa yang seimbang.

## 5.2 Saran

Penelitian lebih lanjut yang dapat dilakukan mengenai masalah deteksi penyakit ginjal kronis adalah:

1. Menggunakan model klasifikasi lain seperti SVM, *Naive-Bayes*, dll. aktivasi ReLu

2. Menambahkan jumlah *dataset* yang digunakan.

3. Menggunakan metode seleksi fitur yang berbeda untuk meningkatkan performa model agar lebih baik lagi.

# Daftar Pustaka


[1] Seely, J. C. B. (2005). Kidney – Introduction Kidney – Introduction. National Toxicology Program, Figure 1, 1–10.

[2] Kidney Research UK. (2010). The Kidneys – a Basic Guide. Kidney Health Information. https://www.nhs.uk/Livewell/Kidneyhealth/Documents/kidney guide.pdf

[3] Kamasita, S. E., Suryono, Nurdian, Y., Hermansyah, Y., Junaidi, E., Mohamat, Fatekurohman. (2018). Pengaruh Hemodialisis Terhadap Kinetik Segmen Ventrikel Kiri Pada Pasien Penyakit Ginjal Kronik Stadium V. NurseLine Journal, 3(1), 11–19.

[4] Almasoud, Marwa E, Tomas. 2019. Detection of Chronic Kidney Disease using Machine Learning Algorithms with Least Number of Predictors. International Journal of Advanced Computer Science and Applications. 10. 10.14569/IJACSA.2019.0100813.

[5] Carney, Ellen F. 2020. The Impact of Chronic Kidney Disease on Global Health. Nature. Diakses pada 17 Maret 2021, dari https://www.nature.com/articles/s41581-020-0268-7: :text=In%202017%2C%20CKD%20resulted%20in,%25%20of%20all%2Dcause%20mortality

[6] NKF. (2020). Estimated Glomerular Filtration Rate (eGFR) — National Kidney Foundation. 2020. https://www.kidney.org/atoz/content/gfr

[7] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. Nature.

[8] Tabreer T. Hasan* , Manal H. Jasim and Ivan A. Hashim. 2017. Heart Disease Diagnosis System based on Multi-Layer Perceptron neural network and Support Vector Machine http://inpressco.com/wp-content/uploads/2017/10/Paper271842-1853.pdf

[9] B.K. Lavine, T.R. Blank. 2009. 3.18 - Feed-Forward Neural Networks. Comprehensive Chemometrics pages 571-586, ISBN 9780444527011. Diakses pada 23 Maret 2021, dari https://doi.org/10.1016/B978-044452701-1.00026-0

[10] Boiko, Sergii. 2018. Practical Machine Learning for Solving Real World Problems. Railsware. Diakses pada 23 Maret 2021, dari https://railsware.com/blog/practical-machine-learning-for-solving-real-world-problems/

[11] Brownlee, Jason. 2020. 4 types of classification tasks in machine learning. Machine Learning Mastery. Diakses pada 11 April 2021 dari https://machinelearningmastery.com/types-of-classification-in-machine-learning/

[12] Repository Dinus. 2011. Pengertian Multi Layer Neural Network. https://repository.dinus.ac.id/docs/ajar/Pengertian_Multi_Layer_Neural_Network.pdf

[13] Shaikh, Raheel. 2018. Feature Selection Techniques in Machine Learning with Python. Towards Data Science. https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:˜:text=Feature%20Selection%20is%20the%20process,learn%20based%20on%20irrelevant%20features.

[14] Paul, Sayak. 2020. Beginner's Guide to Feature Selection in Python. Data Camp.diakses pada 23 April 2021 dari https://www.datacamp.com/community/tutorials/feature-selection-python

[15] James, Gareth. Dkk. 2013. An Introduction to Statistical Learning with Application in R. Springer: USA

[16] Puspitasari, Ana Eka Ratnawati, Dian Wahyu Widodo, Agus. 2018. Klasifikasi Penyakit Gigi Dan Mulut Menggunakan Metode Support Vector Machine. Diakses pada 17 April 2021 dari https://www.researchgate.net/publication/324038271_Klasifikasi_Penyakit_Gigi_Dan_Mulut_Menggunakan_Metode_Support_Vector_Machine

[17] Abushariah, M. A. M., Alqudah, A. A. M., Adwan, O. Y., Yousef, R. M. M. 2014. Automatic Heart Disease Diagnosis System Based on Artificial Neural Network (ANN) and Adaptive Neuro-Fuzzy Inference Systems (ANFIS) Approaches. Journal of Software Engineering and Applications, 07(12), 1055–1064. https://doi.org/10.4236/jsea.2014.712093

[18] Lin, C. H., Yang, P. K., Lin, Y. C., Fu, P. K. 2020. On Machine Learning Models for Heart Disease Diagnosis. 2nd IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability 2020, ECBIOS 2020, 158–161. https://doi.org/10.1109/ECBIOS50299.2020.9203614

[19] Heaton, Jeff. (2017). Heaton Research: The Number of Hidden Layers. Heaton Research. Diakses 10 Juni 2021, dari https://www.heatonresearch.com/2017/06/01/hidden-layers.html

[20] Setyo Nugroho, Kuncahyo. 2019. Confusion Matrix untuk Evaluasi Model pada Supervised Learning. Medium. Diakses pada 23 April 2021, dari https://medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f

[21] Nurhikmat, Triano. 2018. Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek. https://dspace.uii.ac.id/bitstream/handle/123456789/7843/TUGAS%20AKHIR_TRIANO%20NURHIKMAT_14611209_STATISTIKA_UII.pdf?sequence=1

[22] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

# Lampiran

## *Pre-processing*

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
#Import Dataset
dataset = pd.read_excel(r'https://drive.google.com/uc?export=download&id=1zV3EZRkWePzi8wGZLW10RH62snhM5dQH')
dataset.head()
```

```
#Menghitung jumlah missing value di setiap kolom
print(dataset)
print(dataset.isnull().sum())
print(dataset.isnull().sum().sum())
```

```
#Melihat umur keseluruhan pada data
print(dataset['age'].describe())
print(dataset['age'].mode())
dataset['age'].hist()
```

```
# Melihat data keseluruhan hemoglobin
print(dataset['hemo'].describe())
dataset['hemo'].hist()
```

```
#Hubungan anemia dengan hemoglobin (dicek dari data anemia)
data_ane1 = dataset[dataset['ane']=='yes']        #data pengidap anemia
data_ane_hemo1 = data_ane1[['hemo','ane']]        #data pengidap anemia dan hemoglobinnya
data_ane1['hemo'].hist()                          #histogram pengidap anemia
print(data_ane1)
print(data_ane1['hemo'].describe())               #statistik hemoglobin pengidap anemia
print(data_ane_hemo1[data_ane_hemo1['hemo']>12])  #data pengidap anemia namun hemoglobin > 12
data_ane1.isnull().sum()
```

```
#Hubungan anemia dengan hemoglobin (dicek dari data tidak anemia)
data_ane2 = dataset[dataset['ane']=='no']   #data yang tidak anemia
data_ane_hemo2 = data_ane2[['hemo','ane']]  #data hemoglobin yang tidak anemia
print(data_ane_hemo2)
data_ane3 = data_ane2[data_ane2['hemo']<12]     #data yang tidak anemia tetapi hemoglobin < 1
print(data_ane3)
print(data_ane2['hemo'].describe())       #statistik hemoglobin yang tidak anemia
data_ane2['hemo'].hist()      #histogram hemoglobin yang tidak anemia
print(data_ane3[['hemo','age','rbc','ane']])
data_ane2.isnull().sum()
```

```
#Memasukan missing data age dan hemo berdasarkan MEAN
dataset['age'] = dataset['age'].replace(np.nan,51)
dataset['hemo']= dataset['hemo'].replace(np.nan, 12.5)
```

```
#Melihat hubungan Hemoglobin dan PCV (untuk regresi)
dataset_cc = dataset.dropna(axis = 0)
dataset_hemo_pcv = dataset[['hemo','pcv']]
corr1 = dataset_hemo_pcv.corr()
corr1.style.background_gradient(cmap='coolwarm').set_precision(2)
```

```
dataset_hemo_pcv = dataset.dropna(axis = 0, subset = ['hemo','pcv'])
dataset_hemo_pcv = dataset_hemo_pcv.loc[:,['hemo','pcv']]
missing_pcv = dataset['pcv'].isnull()
hemo_mispcv = pd.DataFrame(dataset['hemo'][missing_pcv])
```

```
x1 = dataset_hemo_pcv[['hemo']]
y1 = dataset_hemo_pcv[['pcv']]
from sklearn.model_selection import train_test_split

x1_train, x1_test, y1_train, y1_test = train_test_split(x1,y1, test_size = 0.2, random_state =101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression().fit(x1_train, y1_train)
pcv_pred = lm.predict(hemo_mispcv)
```

```
# Memasukan missing data pcv berdasarkan Regresi dengan Hemo
dataset.loc[dataset['pcv'].isnull(),'pcv'] = pcv_pred
```

```
# Melihat hubungan Hemoglobin dan Red Blood Cell Cost (untuk regresi)
dataset_cc = dataset.dropna(axis = 0)
dataset_hemo_rc = dataset[['hemo','rc']]
corr = dataset_hemo_rc.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

```
dataset_hemo_rc1 = dataset.dropna(axis = 0, subset = ['hemo','rc'])
dataset_hemo_rc1 = dataset_hemo_rc1.loc[:,['hemo','rc']]
missing_rc= dataset['rc'].isnull()
hemo_misrc = pd.DataFrame(dataset['hemo'][missing_rc])
```

```
x = dataset_hemo_rc1[['hemo']]
y = dataset_hemo_rc1[['rc']]
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state =101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression().fit(x_train, y_train)
rc_pred = lm.predict(hemo_misrc)
```

```
# Memasukan missing data rc berdasarkan Regresi dengan Hemo
dataset.loc[dataset['rc'].isnull(),'rc'] = rc_pred
```

```
dataset['htn']=dataset['htn'].replace(np.nan,'no')
data_bpno = dataset[dataset['htn']=='no']
data_bpyes = dataset[dataset['htn']=='yes']
print(data_bpno['bp'].describe())
print(data_bpyes['bp'].describe())
data_bpno['bp'] = data_bpno['bp'].replace(np.nan,70)
data_bpyes['bp'] = data_bpyes['bp'].replace(np.nan,80)
dataset = pd.concat([data_bpno,data_bpyes])
dataset.isnull().sum()
```

```
# Memasukan missing data pcc (notpresent) dan pc (berdasarkan pcc, [notpresent > normal]; [present;abnormal])
dataset['pcc']=dataset['pcc'].replace(np.nan,'notpresent')
data_pcnormal = dataset[dataset['pcc']=='notpresent']
data_pcnormal['pc'] = data_pcnormal['pc'].replace(np.nan,'normal')
data_pcabnormal = dataset[dataset['pcc']=='present']
data_pcabnormal['pc'] = data_pcabnormal['pc'].replace(np.nan, 'abnormal')

dataset = pd.concat([data_pcnormal,data_pcabnormal])
```

```
# Memasukan missing data dm (no), cad(no),appet(good),pe(no),dan ane(no)
dataset['dm']=dataset['dm'].replace(np.nan,'no')
dataset['cad']=dataset['cad'].replace(np.nan,'no')
dataset['appet']=dataset['appet'].replace(np.nan,'good')
dataset['pe']=dataset['pe'].replace(np.nan,'no')
dataset['ane']=dataset['ane'].replace(np.nan,'no')

```

```
datasu_notckd = dataset[dataset['classification']=='notckd']
datasu_ckd = dataset[dataset['classification']=='ckd']

datasu_notckd['su'] = datasu_notckd['su'].replace(np.nan, 0)

dataset = pd.concat([datasu_notckd,datasu_ckd])

data_su0 = dataset[dataset['su']==0]
data_su1 = dataset[dataset['su']==1]
data_su2 = dataset[dataset['su']==2]
data_su3 = dataset[dataset['su']==3]
data_su4 = dataset[dataset['su']==4]
data_su5 = dataset[dataset['su']==5]
data_su6 = dataset[dataset['su'].isnull()]

print(data_su0['bgr'].mean())
print(data_su1['bgr'].mean())
print(data_su2['bgr'].mean())
print(data_su3['bgr'].mean())
print(data_su4['bgr'].mean())
print(data_su5['bgr'].mean())
print(datasu_ckd['bgr'].mean())

data_su0['bgr']=data_su0['bgr'].replace(np.nan,122)
data_su1['bgr']=data_su1['bgr'].replace(np.nan,213)
data_su2['bgr']=data_su2['bgr'].replace(np.nan,256)
data_su3['bgr']=data_su3['bgr'].replace(np.nan,269)
data_su4['bgr']=data_su4['bgr'].replace(np.nan,302)

dataset = pd.concat([data_su0,data_su1,data_su2,data_su3,data_su4,data_su5,data_su6])
dataset['bgr'] = dataset['bgr'].replace(np.nan,172)
```

```
# Melihat mean dari bu, sc, sod, pot, dan sg
print(dataset['bu'].mean())
print(dataset['sc'].mean())
print(dataset['sod'].mean())
print(dataset['pot'].mean())
print(dataset['sg'].mean())
# # Memasukan missing data bu, sc, sod, pot, sg, ba berdasarkan mean
dataset['bu']=dataset['bu'].replace(np.nan, 57)
dataset['sc']=dataset['sc'].replace(np.nan, 3.1 )
dataset['sod']=dataset['sod'].replace(np.nan,138 )
dataset['pot']=dataset['pot'].replace(np.nan, 4.6)
dataset['sg']=dataset['sg'].replace(np.nan,1.017 )
dataset['ba']=dataset['ba'].replace(np.nan,'notpresent' )
```

```
# Melihat data untuk White Blood Cell (wc)
print(dataset['wc'].describe())
dataset['wc'].hist()
# # Memasukan missing data wc berdasarkan median
dataset['wc'] = dataset['wc'].replace(np.nan, 8)
```

```
# Menghilangkan kolom rbc (Red Blood Cell)
dataset = dataset.drop(['rbc'], axis=1)
# Menghilangkan baris yang masih ada missing value
dataset = dataset.dropna(axis=0)
# Melihat missing value yang ada
dataset.isnull().sum()
```

```
# merapihkan beberapa data yang typo dan di cek
dataset['appet']=dataset['appet'].replace('poof','poor')
dataset['appet']=dataset['appet'].replace('goof','good')

print(pd.unique(dataset['appet']))
print(pd.unique(dataset['cad']))
dataset['cad']=dataset['cad'].replace('no ','no')
print(pd.unique(dataset['dm']))
print(pd.unique(dataset['htn']))
print(pd.unique(dataset['pe']))
print(pd.unique(dataset['ane']))
print(pd.unique(dataset['pc']))
print(pd.unique(dataset['pcc']))
print(pd.unique(dataset['ba']))
```

```
# Mengubah data nominal
from sklearn.preprocessing import LabelEncoder
label_1 = LabelEncoder()
dataset['pc'] = label_1.fit_transform(dataset['pc'])
label_2 = LabelEncoder()
dataset['pcc'] = label_2.fit_transform(dataset['pcc'])
label_3 = LabelEncoder()
dataset['ba'] = label_3.fit_transform(dataset['ba'])
label_4 = LabelEncoder()
dataset['htn'] = label_4.fit_transform(dataset['htn'])
label_5 = LabelEncoder()
dataset['dm'] = label_5.fit_transform(dataset['dm'])
label_6 = LabelEncoder()
dataset['cad'] = label_6.fit_transform(dataset['cad'])
label_8 = LabelEncoder()
dataset['pe'] = label_8.fit_transform(dataset['pe'])
label_9 = LabelEncoder()
dataset['ane'] = label_9.fit_transform(dataset['ane'])
label_10 = LabelEncoder()
dataset['appet'] = label_9.fit_transform(dataset['appet'])

dataset['classification']=dataset['classification'].replace('ckd',1)
dataset['classification']=dataset['classification'].replace('notckd',0)
```

```
dataset.sort_values(by=['id'],inplace=True)
dataset.head()
```

```
print(dataset)
```

```
#Export Dataset
from google.colab import files
dataset.to_csv('DatasetPGK(Processed).csv')
files.download('DatasetPGK(Processed).csv')
```

## Seleksi Fitur

```
import pandas as pd

dt = pd.read_csv('https://raw.githubusercontent.com/Syamsyuriani/CKD/main/DatasetPGK(Processed).csv')
```

```
import pandas as pd
import numpy as np
import itertools
import time
import statsmodels.api as sm
import matplotlib.pyplot as plt
from math import exp
```

```
#Inisialisasi variabel
Y = dt.classification
X = dt.drop(columns = ['classification'], axis = 1)
```

```
def processSubset(feature_set):
    model = sm.OLS(Y,X[list(feature_set)])
    regr = model.fit()
    RSS = ((regr.predict(X[list(feature_set)]) - Y) ** 2).sum()
    MSE = RSS / (X.shape[0]-2)
    Cp = (1/X.shape[0]) * (RSS + 2 * len(feature_set) * MSE)
    return {"model":regr, "Cp":Cp}
```

```
def backward(predictors):

    t_a = time.time()

    hasil = []

    for col in itertools.combinations(predictors, len(predictors)-1):
        hasil.append(processSubset(col))

    models = pd.DataFrame(hasil)

    # Pilih model dengan nilai Cp paling kecil
    best_model = models.loc[models['Cp'].argmin()]

    t_b = time.time()
    print("Memproses ", models.shape[0], "model dengan", len(predictors)-1, "predictors dalam", (t_b-t_a), "detik.")
    return best_model
```

```
model_bwd = pd.DataFrame(columns=["Cp", "model"], index = range(1,len(X.columns)))

t_a = time.time()
predictors = X.columns

while(len(predictors) > 1):
    model_bwd.loc[len(predictors)-1] = backward(predictors)[0]
    predictors = model_bwd.loc[len(predictors)-1]["model"].model.exog_names

t_b = time.time()
print("\nTotal waktu yang dibutuhkan:", (t_b-t_a), "detik.")
```

```
def backward(predictors):

    hasil = []

    for col in itertools.combinations(predictors, len(predictors)-1):
        hasil.append(processSubset(col))

    models = pd.DataFrame(hasil)

    # Pilih model dengan nilai Cp paling kecil
    best_model = models.loc[models['Cp'].argmin()]

    return best_model
```

```
model_bwd = pd.DataFrame(columns=["Cp", "model"], index = range(1,len(X.columns)))

predictors = X.columns
while(len(predictors) > 1):
    model_bwd.loc[len(predictors)-1] = backward(predictors)
    predictors = model_bwd.loc[len(predictors)-1]["model"].model.exog_names

    print('\nModel yang dipilih dengan {} variabel:'.format(len(predictors)))
    print(pd.DataFrame({'variable': predictors, 'Koefisien': list(model_bwd.loc[len(predictors), "model"].params)}))
```

```
fitur23 = ['age', 'bp', 'sg', 'al', 'su', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc',
           'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad', 'appet',
           'pe', 'ane']
```

```
pd.set_option('max_colwidth', None)
hasil = pd.DataFrame(columns=["Cp", "model"], index = range(1,len(X.columns)))
predictors = X.columns
model23 = backward(predictors)
fitur = []

while(len(predictors) > 1):
    hasil.loc[len(predictors)-1] = backward(predictors)
    predictors = hasil.loc[len(predictors)-1]["model"].model.exog_names
    fitur.append(predictors)
fitur.reverse()
fitur.append(fitur23)
RS = list(hasil['Cp'])
RS.append(model23['Cp'])
p = pd.DataFrame({'Cp': RS,'Fitur': fitur},index=range(1,24))
p
```

```
p.loc[p['Cp'].argmin() + 1]
```

```
hasil.loc[14]['model'].params
```

## Persiapan Data

```
#persiapan data
dt = pd.read_csv('https://raw.githubusercontent.com/Syamsyuriani/CKD/main/DatasetPGK(Processed).csv')

X = dt.drop(columns=['age', 'su', 'pc', 'pcc', 'ba', 'sod', 'pot', 'appet', 'pe', 'classification'])
X
```

```
#pisah data numerik
var_kategorik = ['al', 'htn', 'dm', 'cad', 'ane']
X_numerik = X.drop(columns=var_kategorik)
X_kategorik = X[var_kategorik]
```

```
#Standarisasi yang numerik
scaler = StandardScaler()
scaler.fit(X_numerik)
X_numerik = scaler.transform(X_numerik)
```

```
#gabung X_numerik & X_kategorik
X_numerik = pd.DataFrame(X_numerik)
X = pd.concat([X_numerik, X_kategorik], axis=1)
X.rename(
    columns={ 0: 'bp', 1: 'sg', 2: 'bgr', 3: 'bu', 4: 'sc', 5: 'hemo', 6: 'pcv', 7: 'wc', 8: 'rc'}),
    inplace=True,
)
```

```
y= dt.iloc[:,-1]

def_encoder = LabelEncoder()
y = def_encoder.fit_transform(y)
```

```
#pisah data train & data test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42)
```

## *Fitting* Model

```
random.seed(69)

def create_model(learn_rate, hidden_layers, neurons):
  model = Sequential()
  model.add(Dense(14))
  for i in range(hidden_layers):
    model.add(Dense(neurons, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  optimizer = SGD(learning_rate=learn_rate)
  model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=['accuracy'])
  return model

model = KerasClassifier(build_fn=create_model, epochs=10, batch_size=32, verbose=0)

param_grid = {"hidden_layers": [1,2,3,4,5], "neurons": [1,2,3,5,6,7,8,9,10,11,12,13,14], "learn_rate": [0.0001, 0.001, 0.01, 0.1, 1]}
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=10)
grid_result = grid.fit(X_train, y_train)

print("Akurasi terbaik: %f menggunakan %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("Akurasi: %f (%f) dengan: %r" % (mean, stdev, param))
```

## Evaluasi Model

```
print(grid_result.best_params_)
```

```
y_pred= grid_result.predict(X_test)
print('Akurasi:',metrics.accuracy_score(y_pred,y_test))
print('Presisi:',metrics.precision_score(y_pred,y_test))
print('Recall:',metrics.recall_score(y_pred,y_test))
```

```
y_pred= grid_result.predict(X_test)
print('Akurasi:',metrics.accuracy_score(y_pred,y_test))
print('Presisi:',metrics.precision_score(y_pred,y_test))
print('Recall:',metrics.recall_score(y_pred,y_test))
```

```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
(tn, fp, fn, tp)
```

```
akurasi = [None]*10
presisi = [None]*10
recall = [None]*10

for i in range(10):
  dt = pd.read_csv('https://raw.githubusercontent.com/Syamsyariani/CKD/main/DatasetPCK(Processed).csv')
  X = dt.drop(columns=['age', 'su', 'pc', 'pcc', 'ba', 'sod', 'pot', 'appet', 'pe', 'classification'])

  #Pisah data numerik
  var_kategorik = ['al', 'htn', 'dm', 'cad', 'ane']
  X_numerik = X.drop(columns=var_kategorik)
  X_kategorik = X[var_kategorik]

  #Standarisasi yang numerik
  scaler = StandardScaler()
  scaler.fit(X_numerik)
  X_numerik = scaler.transform(X_numerik)

  #gabung X_numerik & X_kategorik
  X_numerik = pd.DataFrame(X_numerik)
  X = pd.concat([X_numerik, X_kategorik], axis=1)
  X.rename(
    columns={ 0: 'bp', 1: 'sg', 2: 'bgr', 3: 'bu', 4: 'sc', 5: 'hemo', 6: 'pcv', 7: 'wc', 8: 'rc'},
    inplace=True,
  )

  y= dt.iloc[:,-1]
  def_encoder = LabelEncoder()
  y = def_encoder.fit_transform(y)

  #pisah data train & data test
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

  model = Sequential()
  model.add(Dense(14))
  for j in range(1): ##################### jumlah hidden layer ####################
    model.add(Dense(2, activation='relu')) ###################### jumlah neuron ########################
  model.add(Dense(1, activation='sigmoid'))
  optimizer = SGD(learning_rate=1) ################## learning rate ######################
  model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=['accuracy'])

  model.fit(X_train, y_train, batch_size=32, epochs = 10, validation_data = (X_train, y_train))

  y_pred = model.predict_classes(X_test)
```

```
  TP = 0
  FP = 0
  TN = 0
  FN = 0
  for k in range(len(y_pred)):
    if y_pred[k] == 1 and y_test[k] == 1:
      TP += 1
    if y_pred[k] == 1 and y_test[k] == 0:
      FP += 1
    if y_pred[k] == 0 and y_test[k] == 0:
      TN += 1
    if y_pred[k] == 0 and y_test[k] == 1:
      FN += 1

  akurasi[i] = (TP + TN) / len(y_pred)
  presisi[i] = (TP) / (TP + FP)
  recall[i]= (TP) / (TP + FN)
```

```
print('Rata-rata akurasi untuk 10 kali pembelajaran: ' + str(np.mean(akurasi)))
print('Rata-rata presisi untuk 10 kali pembelajaran: ' + str(np.mean(presisi)))
print('Rata-rata recall untuk 10 kali pembelajaran: ' + str(np.mean(recall)))
```

```
from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import SGD

def create_model(hidden_layers, learn_rate, neurons):
  model = Sequential()
  model.add(Dense(14))
  for i in range(hidden_layers):
    model.add(Dense(neurons, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  optimizer = SGD(learning_rate=learn_rate)
  model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=['accuracy'])
  return model

model = KerasClassifier(build_fn=create_model, epochs = 10, batch_size = 32, verbose=0)

param_grid = {'hidden_layers': [1], 'learn_rate': [1], 'neurons': [2]}
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=10, scoring='accuracy',return_train_score=True)
```

```
grid.fit(X_train,y_train)
grid.cv_results_
```

```
scores = cross_val_score(grid, X_train, y_train, cv=10)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```