Jurnal Kajian dan Terapan Matematika
Volume 10, Edisi 3, Bulan November 2024, 200 - 214
http://journal.student.uny.ac.id/ojs/index.php/jktm

MODEL SISTEM ANTRIAN DAN ANALISIS PELAYANAN PASIEN PADA LOKET PENGAMBILAN OBAT DI PUSKESMAS (STUDI KASUS: PUSKESMAS KALASAN)

QUEUE SYSTEM MODEL AND PATIENT SERVICE ANALYSIS AT DRUG COLLECTION COUNTRIES AT PUSKESMAS (CASE STUDY: KALASAN HEALTH CENTER)

Alya Shalahuddin Akbar, Prodi Matematika FMIPA UNY
Muhammad Fauzan*, Prodi Matematika FMIPA UNY
*e-mail: mfauzan@uny.ac.id

Abstrak

Tujuan penelitian ini adalah mendeskripsikan sistem antrian pada loket pengambilan obat dan mendeskripsikan jumlah fasilitas pelayanan yang optimal pada saat kondisi ramai bagian loket pengambilan obat di Puskesmas Kalasan. Penelitian ini merupakan penelitian terapan dengan menggunakan analisis teori antrian. Populasi pada penelitian ini yaitu semua resep obat yang dikeluarkan oleh dokter dan resep yang masuk dalam antrian pada Puskesmas Kalasan. Sampel pada penelitian ini yaitu resep obat yang masuk antrian pada loket pengambilan obat di Puskesmas Kalasan saat kondisi ramai. Data dikumpulkan dengan lembar observasi. Hasil penelitian menunjukkan bahwa model antrian yang diterapkan di Puskesmas Kalasan pada loket pengambilan obat yaitu *(M/M/3):(FCFS/∞/∞)* dengan sistem antrian *Multi Channel Single Phase.* Sistem pelayanan di loket pengambilan obat Puskesmas Kalasan pada umumya sudah baik atau optimal, walaupun pada hari tertentu fasilitas pelayanan masih ada yang belum optimal. Hal tersebut ditunjukan dengan nilai tingkat kegunaan fasilitas yang secara umum masih di bawah 66%, yaitu sebesar 63,3%.

Kata kunci: model sistem antrian, antrian loket pengambilan obat, analisis pelayanan, Puskesmas Kalasan.

Abstract

The aims of this study were to describe the queuing system at the drug collection counter and describe the optimal number of service facilities when conditions are crowded at the drug collection counter at the Kalasan Health Center. This research is an applied research using queuing theory analysis. The population in this study were all prescription drugs issued by doctors and prescriptions that were included in the queue at the Kalasan Health Center. The sample in this study were drug prescriptions that entered the queue at the drug collection counter at the Kalasan Health Center when conditions were crowded. Data was collected with observation sheets. The results showed that the queuing model applied at the Kalasan Health Center at the drug collection counters was (M/M/3):(FCFS/∞/∞) with a Single Phase Multi Channel queuing system. The service system at the Kalasan Health Center drug collection counter is generally good or optimal, although on certain days there are still service facilities that are not optimal. This is indicated by the level of usefulness of the facility which is generally still below 66%, which is 63.3%.
Keywords*: SEIR Model, NCT physical album.*

PENDAHULUAN

Penduduk merupakan orang atau sekumpulan orang yang tinggal di suatu daerah dengan mengikuti semua aturan yang telah di tentukan. Jumlah penduduk di suatu negara maupun wilayah mengalami peningkatan setiap tahunnya. Menurut (Badan Pusat Statistik, 2017), DIY merupakan salah satu provinsi dengan jumlah penduduk yang cukup besar yaitu sebesar 3.669.200 jiwa pada tahun 2015. Jumlah penduduk juga merupakan faktor mendasar yang mempengaruhi banyak hal, seperti tingkat kemiskinan, fasilitas, dan ekonomi. Karena jumlah penduduk di DIY cukup besar, maka tingkat kemiskinan juga akan semakin besar. Pada tahun 2011 – 2021, DIY merupakan wilayah dengan tingkat kemiskinan yang paling tinggi dibandingkan dengan wilayah lainnya yaitu menempati urutan pertama dan disusul oleh daerah lain seperti Jawa Tengah, Jawa Timur, Jawa Barat, Banten, dan DKI Jakarta.

Melihat dari tingkat kemiskinan yang tidak stabil dan masih di atas 10%, kemiskinan merupakan masalah yang harusnya dipandang sebagai masalah sosial yang kompleks, sehingga tingkat kemiskinan perlu diprediksi agar pemerintah dapat mengambil keputusan dalam menanggulangi masalah kemiskinan. Untuk memprediksi kemiskinan, peneliti menggunakan beberapa faktor yang diduga berpengaruh seperti jumlah penduduk, jumlah pengangguran, dan nilai tukar petani. Jumlah penduduk sangat berpengaruh pada kemiskinan, karena jika perkembangan penduduk lebih cepat dibandingkan hasil produksi pertanian maka akan menyebabkan penduduk kesulitan dalam memenuhi kebutuhan hidup, sehingga akan berimbas pada meningkatnya kemiskinan. Selain jumlah penduduk pengangguran merupakan salah satu faktor yang mempengaruhi kemiskinan. Meningkatnya pengangguran memiliki dampak yang memberatkan pada ketimpangan pendapatan. Faktor lain yang mempengaruhi kemiskinan adalah NTP. Jika NTP > 100 maka akan memiliki dampat penurunan terhadap kemiskinan di pedesaan, sebaliknya jika NTP < 100 maka akan memiliki dampak peningkatan terhadap kemiskinan di pedesaan (Badan Pusat Statistik, 2017).

Metode yang dipandang tepat untuk memprediksi tingkat kemiskinan adalah algoritma *Artificial Neural Network (ANN)* atau Jaringan Syaraf Tiruan (JST), yang merupakan salah satu teknik klasifikasi yang cukup handal dikarenakan kemampuannya dalam memprediksi dan algoritmanya dapat meniru prinsip kerja dari jaringan syaraf manusia. *Backpropogation*, merupakan salah satu metode dari JST yang memiliki keseimbangan jaringan untuk memberikan respon balik terhadap pola masukan. *Backpropagation* dalam cara kerjanya meggunakan memori yang lebih sedikit dari metode lainnya dan memberikan hasil dengan kecepatan pemrosesan yang cukup cepat dengan tingkat kesalahan yang masih dapat diterima (Aprizal et al., 2019). Beberapa penelitian yang terkait dengan *Backpropagation* yaitu pada penelitian Khusniyah & Sutikno (2016) mengenai prediksi nilai tukar petani yang diperoleh akurasi hasil sebesar 99,39%. Penelitian lain mengenai *Backpropagation* ditulis oleh Sunil Setti, Anjar Wanto (2018) mengenai prediksi jumlah pengguna internet yang diperoleh akurasi sebesar 92%. Berdasarkan penelitan yang telah dilakukan, maka dapat diketahui bahwa metode *Backpropagation* menghasilkan akurasi yang cukup tinggi dan baik dalam masalah prediksi.

Penelitian ini bertujuan untuk memberikan model prediksi kemiskinan yang diacu dari jumlah penduduk, jumlah pengangguran, dan nilai tukar petani dengan akurasi yang lebih baik dengan menggunakan JST. Tujuan lain dari penelitian ini adalah memberikan informasi kepada masyarakat khususnya bagi Badan Kependudukan tingkat kemiskinan dapat diprediksi menggunakan algoritma JST *Backpropagation*.

METODE

Penelitian ini menggunakan metode yang diperuntukan pada jenis data yang saling berkaitan. Metode yang digunakan adalah Jaringan Syaraf Tiruan (JST) *Backpropogation*. Penelitian ini dilakukan di Provinsi Daerah Istimewa Yogyakarta. Data yang digunakan keseluruhan berasal dari website resmi BPS (https://yogyakarta.bps.go.id/). Data penelitian

yang digunakan berupa tingkat kemiskinan, jumlah penduduk, jumlah pengangguran, dan NTP di Provinsi DIY tahun 2011 - 2021. Penelitian ini menggunakan analisis pengolahan data sesuai dengan kerangka dan fitur menggunakan *Backpropagation* sebagai fokus dari penelitian ini. Langkah penelitian ini terdiri dari tahapan : Pembagian Data (*Training* dan *Testing*), Menentukan Model Terbaik (Fase I = *Feedforward*, Fase II = *Backpropagation*, Fase III = Perubahan Bobot), Menguji Model Prediksi, dan Mengevaluasi Akhir. Pada proses pengolahan data peneliti menggunakan bantuan MATLAB dari awal hingga akhir. Secara lebih jelas, berikut penjabaran dari langkah – langkah tersebut.

1. Pembagian Data

   Pada tahap ini akan dilakukan pemiilihan data sebagai data input dan testing terhadap 11 data yang berisi angka tingkat kemiskinan, jumlah penduduk, jumlah pengangguran, dan NTP. Aspek pembagian data harus ditekankan untuk memperoleh data *training* yang cukup dan dapat digunakan untuk proses pembelajaran. Berbeda dengan data *training*, data *testing* digunakan untuk menguji proses pembelajaran yang dilakukan oleh data *training* berdasarkan nilai *MAPE*.
2. Menentukan Model Terbaik
   a. Fase I = *Feedforward*

      Feedforward merupakan algoritma yang memakai variabel masukan untuk mempengaruhi variabel masukan lain dalam sistem. Algoritma ini juga merupakan bentuk *Neural Network* yang paling sederhana karena informasi hanya diproses dalam satu arah. Data dapat melewati beberapa *hidden node*, namun selalu bergerak dalam satu arah dan tidak pernah mundur ke belakang. Dengan kata lain, jaringan syaraf *feedforward* hanya memungkinkan sinyal untuk melakukan perjalanan melalui satu jalur saja, yakni *input* ke *output*. Tidak ada koneksi umpan balik dari *output* ke dirinya sendiri (*loop*) (Heriyanto, 2016).
   b. Fase II = *Backpropagation*

      *Backpropagation Neural Network (*BPNN*)* merupakan model *NN* dengan *multilayers* yang sering digunakan pada perkiraan data deret berkala. Metode ini juga merupakan metode yang baik dalam menangani masalah – masalah pengenalan pola – pola komples.
   c. Fase III = Perubahan Bobot

      Pada fase ini, proses pembelajaran terhadap perbuahan bobot menggunakan metode *supervised learning*. Pada proses pembelajaran, satu pola *input* akan diberikan ke satu neuron pada lapisan *input*. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan *output*. Apabila terjadi perbedaan antara pola *output* hasil pembelajaran dengan pola target, maka akan muncul *error*. Apabila nilai *error* cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi (Data et al., 2014).
3. Menguji Model Prediksi

   Pada tahap ini merupakan proses mulainya prediksi menggunakan algoritma model terbaik yang telah diperoleh dari proses sebelumnya. Oleh karena itu, tahap ini memiliki sub proses sebagai berikut.
   a. Mencari z_net_j, dengan rumus

$$z\_net_j = v_{j0} + \sum_{i=1}^{3} x_i v_{ji}$$

   b. Mencari z_j, dengan rumus

$$z_j = f(z\_net_j) = \frac{1-e^{-z\_net_j}}{1+e^{-z\_net_j}}$$

c. Mencari y_net_k, dengan rumus

$$y\_net_k = w_{k0} + \sum_{j=1}^{15} w_{kj}.z_j$$

d. Mencari y_k, dengan rumus

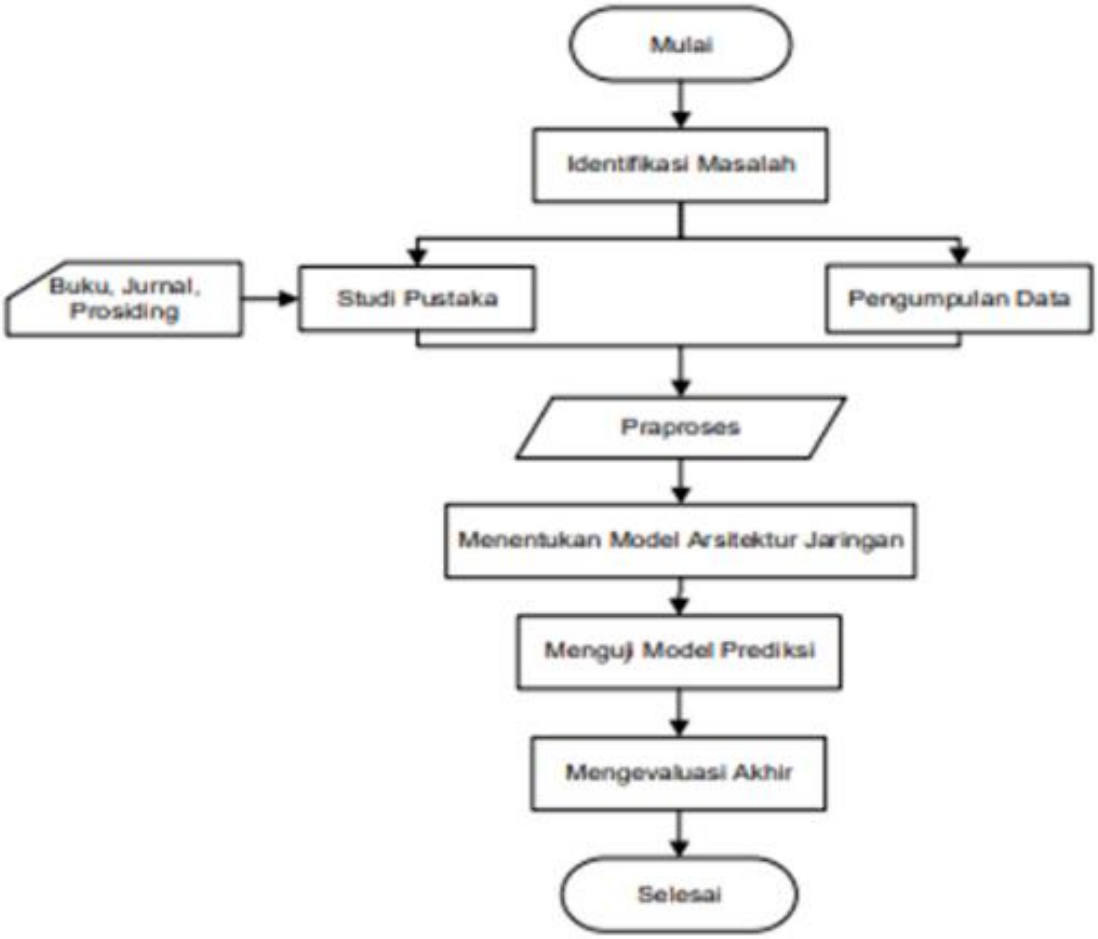
$$y_k = y\_net_k$$

e. Denormalisasi nilai y_k agar menjadi hasil prediksi

4. Mengevaluasi Akhir

Pada tahap ini akan dilakukan interpretasi hasil yang diperoleh dengan memunculkan hasil prediksi, *MAPE*, dan tingkat akurasi.

Gambar 1 menunjukkan keseluruhan diagaram alur dari seluruh proses dalam melakukan penelitian ini.



Gambar 1. Diagram alur penelitian

HASIL DAN PEMBAHASAN

Pembelajaran *Backpropagation Neural Network* (BPNN)

1. Algoritma BPNN

Input dalam BPNN melewati lapisan–lapisan sampai hasil akhir dihitung dan dibandingkan dengan output nyata untuk menemukan *error* nya. *Error* ini kemudian disebarkan kembali ke input untuk menyesuaikan bobot dan bias dalam setiap lapisan. Terdapat dua parameter yang dapat mempercepat pembelajaran algoritma BPNN yaitu, (Priambodo et al., 2020).

a. *Learning rate* (Laju Pembelajaran)

Semakin besar *learning rate* akan berimplikasi pada besarnya langkah pembelajaran sehingga algoritma menjadi tidak stabil. Sebaliknya jika *learning rate* di set terlalu kecil maka algoritma akan konvergen dalam jangka waktu yang sangat lama.

b. Momentum

Momentum juga dapat mempercepat pembelajaran dari BPNN. Dalam momentum terjadi perubahan bobot yang merupakan kombinasi dari gradien saat ini dengan gradien sebelumnya. Hal ini menguntungkan di saat suatu data sangat berbeda dengan data lainnya untuk menjaga pelatihan agar berlangsung cepat.

Algoritma BPNN untuk jaringan dengan satu lapisan *hidden layer* menggunakan fungsi aktivasi *sigmoid bipolar* adalah sebagai berikut.

Langkah 0 : Inisiasi bobot dengan mengambil bobot awal menggunakan nilai random terkecil

Langkah 1 : Menetapkan Parameter

a. Maksimum *Epoch*
Maksimum *epoch* adalah jumlah maksimum iterasi yang ditetapkan

b. Target *error*
Target *error* adalah batasan toleransi *error* yang disajikan

c. *Learning Rate (α)*
Learning rate adalah laju pembelajaran yang semakin besar *learning rate* tersebut, maka akan berimplikasi pada besar langkah pembelajaran

Langkah 2 : Inisiasi *epoch* = 0, *MSE* = 1

Fase I : *Feedforward*

Langkah 3 : Tiap – tiap unit di lapisan *input* $(x_i, i = 1,2,\dots,n)$ menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit yang ada di *hidden layer*

Langkah 4 : Tiap – tiap unit pada *hidden layer* $(z_j, j + 1,2,\dots,p)$ menjumlahkan sinyal – sinyal *input* berbobot

$$z_{net_j} = v_{j0} + \sum_{i=1}^{n} x_i v_{ji}$$

$$z_j = f\left(z_{net_j}\right) = \frac{1 - e^{-z\_net_j}}{1 + e^{-z\_net_j}}$$

Langkah 5 : Tiap – tiap unit *output* $(y_k, k = 1,2,\dots,m)$ menjumlahkan sinyal – sinyal *input* berbobot

$$y_{net_x} = w_{k0} + \sum_{j=1}^{p} z_j w_{kj}$$

$$y_k = y\_net_k$$

Fase II : *Backpropagation*

Langkah 6 : Tiap – tiap *output* $(y_k, k = 1,2,\dots,m)$ menerima target pola yang berhubungan dengan pola *input* pembelajaran dengan *error*

$$\delta_k = (t_k - y_k) f'\left(y_{net_k}\right) = (t_k - y_k) y_k (1 - y_k)$$

δ_k merupakan unit *error* yang dipakai dalam perubahan bobot lapis bawahnya (Langkah 7). Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α.

$$\Delta w_{kj} = \alpha.\delta_k.z_j\ ;(k = 1,2,\dots,m; j = 1,2,\dots,p)$$

Langkah 7 : Tiap – tiap unit tersembunyi $(z_j, j = 1,2,\dots,p)$ menjumlahkan hasil perubahan *input* nya dari unit – unit dilapisan atasnya

$$\delta_{net_j} = \sum_{k=1}^{m} \delta_k w_{kj}$$

Faktor δ unit tersembunyi, yaitu

$$\delta_j = \delta_{net_j}.f'\left(z_{net_j}\right) = \delta_{net_j}.z_j(1 - z_j).$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji})

$$\Delta v_{ji} = \alpha.\delta_j.x_i\ ;(j = 1,2,\dots,p; i = 1,2,\dots,n)$$

Fase III : Perubahan bobot

Langkah 8 : Hitung semua perubahan bobot. Perubahan bobot yang menuju ke *output* unit yaitu

$$w_{kj(baru)} = w_{kj(lama)} + \triangle w_{kj}\ ;(k = 1,2,\dots,m; j = 1,2,\dots,p).$$

Perubahan bobot garis menuju ke *hidden layer* adalah

$$v_{ji(baru)} = v_{ji(lama)} + \triangle v_{ji}\ ;(j = 1,2,\dots,p; i = 1,2,\dots,n).$$

Langkah 9 Selesai

Model *BPNN* secara matematis dapat dituliskan sebagai

$$y_k = \sum_{j=1}^{p} w_{kj}.f\left[v_{jo} + \sum_{i=1}^{n} x_i v_{ji}\right] + w_{k0}$$

dengan *f* merupakan fungsi aktivasi *sigmoid bipolar*

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

Berdasarkan algoritma di atas dapat diketahui bahwa pelatihan BPNN meliputi 3 fase sebagai berikut.

a. Fase I: *Feedforward*

Dalam *feedforward* sinyal *input* (x_i) dipropagasikan ke *hidden layer* menggunakan fungsi aktivasi *sigmoid bipolar*. *Output hidden layer* (z_j) tersebut dipropagasikan maju ke lapisan *hidden layer* diatasnya menggunakan fungsi aktivasi yang selanjutnya menghasilkan *output* (y_k). *Output* jaringan (y_k) dibandingkan dengan tagert (t_k). Selisih $t_k - y_k$ merupakan *error* yang terjadi. Jika *error* ini lebih kecil dari batas toleransi yang ditentukan, maka iterasi berhenti. Akan tetapi jika *error* masih lebih besar dari batas toleransinya, maka bobot setiap garis dalam jaringan akan dimodifikasikan untuk mengurangi *error* yang terjadi.

b. Fase II: *Backpropagation*

Berdasarkan *error* $t_k - y_k$, dihitung faktor $\delta_k(k = 1,2,\dots,m))$ yang dipakai untuk mendistribusikan *error* di unit y_k ke semua unit tersembunyi yang terhubung langsung dengan y_k. δ_k juga dipakai untuk mengubah bobot garis yang menghubungkan langsung dengan unit *output*. Dengan cara yang sama, dihitung δ_j

di setiap unit di *hidden layer* sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di lapis bawahnya. Demikian seterusnya hingga faktor δ di unit tersembunyi yang berhubungan lngsung dengan unit masukan dihitung.

c. Fase III: Perubahan Bobot

Setelah semua faktor δ dihitung, bobot semua garis dimodifikasi bersamaan. Perubahan bobot suatu garis didasarkan atas faktor δ neuron di lapis atasnya. Perubahan bobot garis yang menuju ke *output layer* didasarkan atas dasar δ_k yang ada di unit *output*. Ketiga fase tersebut diulang – ulang terus hingga kondisi penghentian dipenuhi. Umumnya kondisi penghentian yang sering dipakai adalah jumlah iterasi atau *error*. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditetapkan, atau jika *error* yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan.

2. Membangun BPNN

Pada dasarnya untuk membangun sebuah jaringan diawali dengan menentukan *input* jaringan. Selanjutnya mengestimasi banyak neuron yang terletak pada lapisan tersembunyi. Pada Matlab, untuk membangun jaringan BPNN dengan menggunakan fungsi *newff*, yaitu

$$\boldsymbol{net = newff(PR, [S1\ S2 \dots SN1], \{TF1\ TF2 \dots TFN1\}, BTF, BLF, PF)}$$

dengan

PR :Matriks berukuran Rx2 yang berisi nilai minimum dan maksimum, dengan R adalah jumlah variabel *input*
Si :Jumlah neuron pada lapisan ke – i, dengan i = 1,2,…,N1
TFi :Fungsi aktivasi pada lapisan ke – i, dengan i = 1,2,…,N1 (default: *logsig*)
BTF :Fungsi pelatihan jaringan (default: *traingdx*)
BLF :Fungsi pelatihan untuk bobot (default: *learngdm*)
PF : Fungsi kinerja (default: *mse*)

Fungsi aktivasi *TFi* harus merupakan fungsi yang dapat dideferensikan, seperti *tansig*, *logsig*, atau *purelin*. Fungsi pelatihan *BTF* dapat digunakan fungsi – fungsi pelatihan *backpropagation*, seperti *traingd*, *traingdm*, *traingdx*, *traingda*, *trainrp*, *traincgf*, *traingcgb*, *traingscg*, *traingbfg*, *trainoss*, dan *trainlm*. (Azid et al., 2017).

3. Pembelajaran BPNN

Selain dengan memasukan sumber atau informasi yang ada pada lapisan *input* dengan membagi data menjadi 2 yaitu data *training* dan data *testing*. Terdapat beberapa langkah yang harus dilakukan untuk mendapatkan model BPNN yang terbaik.

a. Normalisasi Data

Sebelum melakukan pembelajaran maka data perlu dinormalisasikan. Hal ini dapat dilakukan dengan meletakan data – data *input* dan target pada *range* tertentu. Proses normalisasi dapat dilakukan dengan bantuan *mean* dan standar deviasi.

1) Perhitungan nilai rata – rata

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_t$$

2) Perhitungan nilai varians

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

3) Perhitungan normalisasi

$$norm = \frac{x_t - \bar{x}}{\sigma}$$

Normalisasi data dengan bantuan mean dan standar deviasi menggunakan perintah *prestd* pada Matlab yang akan membawa data ke dalam bentuk normal. Berikut perintah *prestd* pada Matlab

[pn,meanp,stdp,tn,meant,stdt]=prestd(P',T')

dengan

P' : Transpose dari matriks *input*
T' : Transpose dari matriks *target*

Fungsi Matlab akan menghasilkan

pn : Matriks *input* yang ternormalisasi
tn : Matriks *target* yang ternormalisasi
meanp : *Mean* pada matriks input asli (p)
stdp : Deviasi standar pada matriks input asli (p)
meant : *Mean* pada matriks target asli (t)
stdt : Deviasi standar pada matriks target asli (t)

b. Estimasi bobot BPNN

Sebelum melakukan estimasi bobot pada BPNN dilakukan terlebih dahulu normalisasi pada keseluruhan data baik data *training* maupun data *testing*. Pembelajaran pada jaringan BPNN ini dilakukan untuk melakukan pengaturan bobot agar diperoleh bobot yang baik. Bobot – bobot tersebut dapat meminimalkan fungsi kinerja jaringan selama pembelajaran berlangsung. Salah satu algoritma gradient descent pada program Matlab adalah *batch mode*. *Batch mode* merupakan penghitungan *gradient* dan perbaikan nilai bobot – bobot yang dilakukan setelah pengoperasian semua *input* data. Pembelajaran BPNN dengan *batch mode* digunakan dengan fungsi *train*.

[net,tr]=train(net,P,T,Pi,Ai)

dengan

net :Jaringan syaraf
tr :Informasi pelatihan (*epoch* dan fungsi kinerja)
P :Matriks data *input*
T :Matriks data *target* (*default: 0*)
Pi :Kondisi awal *delay input* (*default: 0*)
Ai :Kondisi awal *delay* lapisan (*default: 0*)

Parameter yang perlu diset untuk pembelajaran ini adalah fungsi pelatihan menjadi *traingdx*. Parameter – parameter yang harus diset untuk pelatihan ini ada sekitar 5 parameter.

1) Maksimum epoch

Maksimum *epoch* adalah jumlah *epoch* maksimum yang boleh dilakukan selama proses pelatihan. Iterasi akan dihentikan apabila nilai *epoch* melebihi maksimum *epoch*.

net.trainParam.epochs=MaxEpoch

2) Kinerja Tujuan

Kinerja tujuan adalah *target* nilai fungsi kinerja. Iterasi akan dihentikan apabila nilai fungsi kinerja kurang dari atau sama dengan kinerja tujuan.

net.trainParam.goal=TargetError

3) *Learning Rate (α)*

Learning rate adalah laju pembelajaran. Semakin besar nilai *learning rate* maka akan berimplikasi pada semakin besarnya langkah

pembelajaran. Semakin besar nilai α, maka akan semakin cepat proses pelatihan. Akan tetapi jika α terlalu besar, maka algoritma menjadi tidak stabil dan mencapai titik minimum lokal. Sebaliknya, jika *learning rate* diset terlalu kecil maka algoritma akan konvergen dalam jangka waktu yang sangat lama.

net.trainParam.Ir=LearningRate

4) Momentum

Momentum adalah perubahan bobot yang didasarkan atas arah *gradient* pola terakhir dan pola sebelumnya. Besarnya momentum antara 0 sampai 1. Apabila momentum = 0, maka perubahan bobot hanya akan dipengaruhi oleh gradientnya. Tetapi, apabila nilai momentum = 1, maka perubahan bobot akan sama dengan perubahan bobot sebelumnya.

net.trainParam.mc=Momentum

5) Maksimum kenaikan kinerja

Maksimum kenaikan kinerja adalah nilai maksimum kenaikan error yang diijinkan, antara error saat ini dan error sebelumnya.

net.trainParam.max_perf_inc=MaxPerfInc

c. Denormalisasi

Denormalisasi data adalah untuk mengembalikan data pada bentuk semula karena data yang telah di normalisasikan. Apabila data yang dinormalisasikan dengan fungsi *prestd* maka data akan didenormalisasi dengan fungsi *poststd*, dengan *syntak* fungsi pada Matlab:

[P,T]=poststd(pn,meanp,stdp,to,meant,stdt).

Data yang telah di normalisasi menghasilkan *output* pada jaringan syaraf dengan rata – rata (*mean*) = 0 dan *standar deviasi* = 1. Data yang disimulasikan pada jaringan syaraf tiruan juga perlu untuk didenormalisasikan dengan cara yang sama yaitu dengan fungsi *poststd*. *Syntak* pada Matlab sebagai berikut.

an=sim(net,pn) (simulasi jaringan syaraf tiruan)

a=poststd(an,meant,stdt)

Jika pada jaringan syaraf yang telah dilatih menggunakan fungsi *prestd* untuk preprocessing, maka jika terdapat *input* baru yang akan disimulasikan juga harus disesuaikan dengan *mean* dan *standar deviasi* pada jaringan tersebut. Permasalahan tersebut dapat diatasi oleh Matlab dengan fungsi trastd. Berikut *syntak* yang diberikan Matlab.

Qn=trastd(tn,meanp,stdp)
bn=sim(net,Qn)
b=poststd(bn,meant,stdt)

dengan

Q	:Data *input* baru
Qn	:Hasil simulasi dari data *input*
bn/an	:*Output* hasil simulasi
b/a	:*Output* hasil simulasi telah didenormalisasi

4. Pemodelan *Backpropagation Neural Network (*BPNN*)*

Langkah – langkah yang dilakukan untuk pemodelan prediksi menggunakan BPNN adalah sebagai berikut (Hijriah & Narang, 2020).

a. Pembagian data *training* dan *testing*

Data yang dibagi menjadi 2 yaitu data *training* dan data *testing*. Beberapa komposisi data *training* dan *testing* yang sering digunakan adalah 80% untuk data *training* dan 20% untuk *testing*, 75% untuk data *training* dan 25% untuk *testing*,

atau 50% untuk data *training* dan 50% untuk *testing*. Komposisi ini bebas dilakukan sesuai dengan data yang akan di olah.

b. Estimasi model

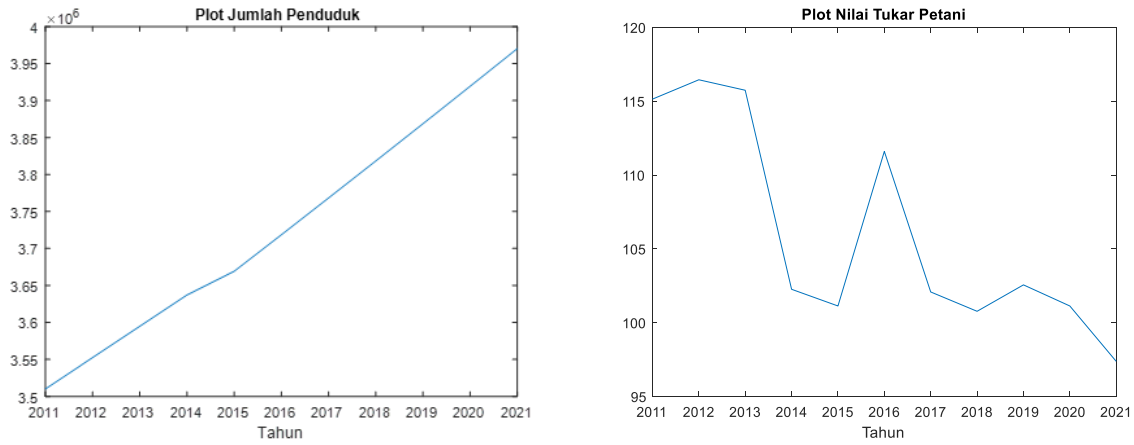
Estimasi model diperoleh melalui hasil pembelajaran pada data *training* dengan membangun model terbaik. Model terbaik diperoleh dengan *trial and error* terhadap beberapa macam arsitektur dengan kriteria arsitektur terbaik, yaitu arsitektur yang menghasilkan nilai *MAPE* terkecil. Estimasi model ini dilakukan dengan menentukan banyak neuron pada *hidden layer* dan membandingkan hasil pembelajaran terbaik dari data *training* dan *testing*. Setelah terbentuknya arsitektur jaringan dari model terbaik, pada hasil pembelajaran akan diperoleh bobot-bobot yang digunakan sebagai parameter pada model jaringan yang terbangun. Bobot-bobot tersebut digunakan untuk meramalkan data periode selanjutnya.

c. Prediksi

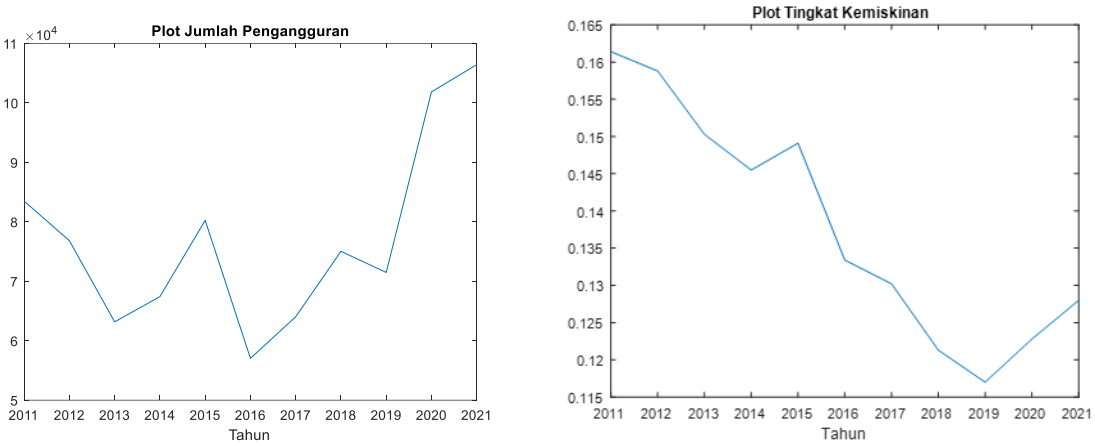
Setelah proses *validasi* dengan menggunakan data *training*, langkah selanjutnya adalah meramalkan data pengamatan berdasarkan struktur jaringan yang telah terbangun. Dengan menggunakan data *target* dari data *training* dan *testing*.

Penerapan BPNN pada Prediksi Tingkat Kemiskinan di DIY

Prediksi ini dilakukan untuk memprediksi tingkat kemiskinan di Daerah Istimewa Yogyakarta. Input yang digunakan merupakan data dari tahun 2011-2021 yang meliputi data jumlah penduduk, jumlah pengangguran, NTP, dan tingkat kemiskinan di Daerah Istimewa Yogyakarta yang peneliti ambil dari website resmi Badan Pusat Statistik DIY (https://yogyakarta.bps.go.id/). Berikut plot dari masing-masing data yang disajikan pada Gambar 2 dan Gambar 3.



Gambar 2. Plot jumlah penduduk dan nilai tukar petani



Gambar 3. Plot jumlah pengangguran dan tingkat kemiskinan

Langkah – langkah prediksi tingkat kemiskinan di DIY menggunakan *BPNN* :

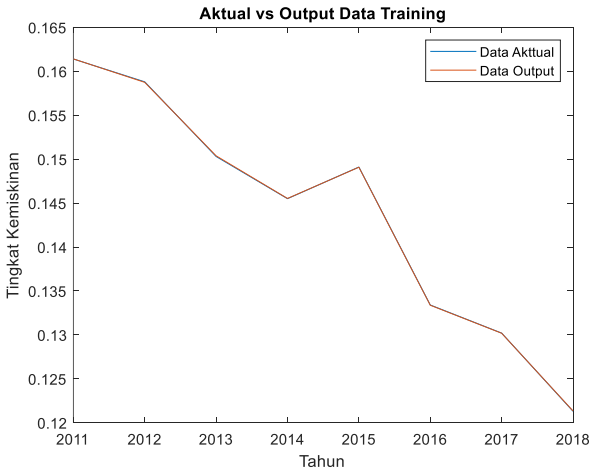
1. Pembagian Data

   Pembagian data untuk prediksi tingkat kemiskinan ini dibagi menjadi dua yaitu data *training* dan data *testing*. Pada penelitian ini digunakan komposisi 73% untuk data *training* dan 27% untuk data *testing*. Pembagian data ini tidak bulat karena data berjumlah 11, sehingga diperoleh 8 data *training* dan 3 data *testing*. *Input* yang digunakan adalah x_1 sebagai data jumlah penduduk, x_2 sebagai data jumlah pengangguran, x_3 sebagai data NTP, dan data tingkat kemiskinan digunakan sebagai target.

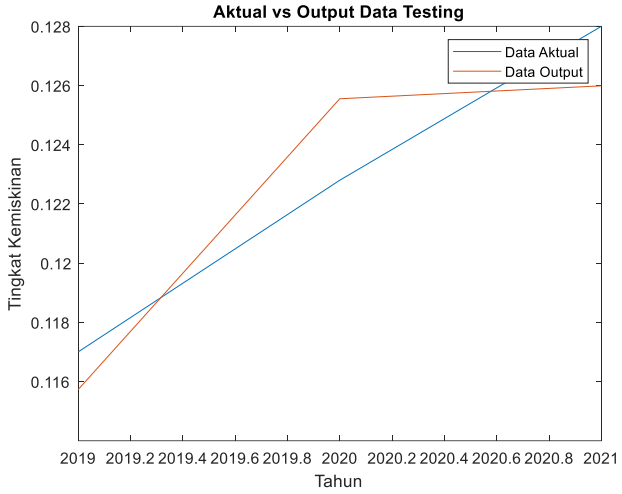
2. Estimasi Model

   Data telah dinormalisasikan dengan perintah *prestd* menggunakan Matlab. Fungsi aktivasi yang digunakan yaitu *sigmoid bipolar (tansig)* pada *hidden layer* dan fungsi aktivasi *linier (purelin)* pada *output layer*. Pembelajaran BPNN menggunakan *traingdx*, dimulai dari arsitektur dengan 4 neuron hingga 15 neuron pada *hidden layer* dengan maksimum 200 iterasi.

   Jaringan dengan 15 neuron pada *hidden layer* akan menjadi arsitektur *BPNN* terbaik karena arsitektur ini menghasilkan nilai *MAPE* terkecil pada data *training* sebesar 0,02414988 dan data *testing* sebesar 1,633300171. Dalam penelitian ini model *BPNN* yang dibangun melalui arsitektur jaringan dengan 15 neuron pada *hidden layer* dengan *input* x_1, x_2, dan x_3 dapat digunakan sebagai model prediksi tingkat kemiskinan di DIY. Berikut akan disajikan plot data tingkat kemiskinan aktual (target) dengan hasil prediksi menggunakan arsitektur jaringan yang terbaik untuk data *training* dan data *testing*.

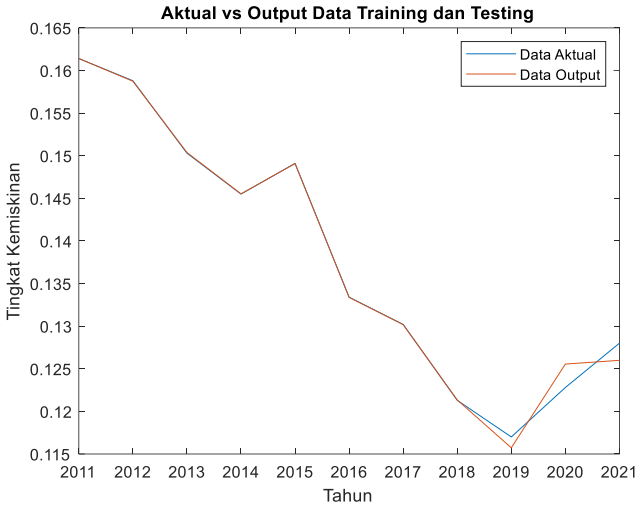


Gambar 4. Plot nilai aktual vs *output* data *training*



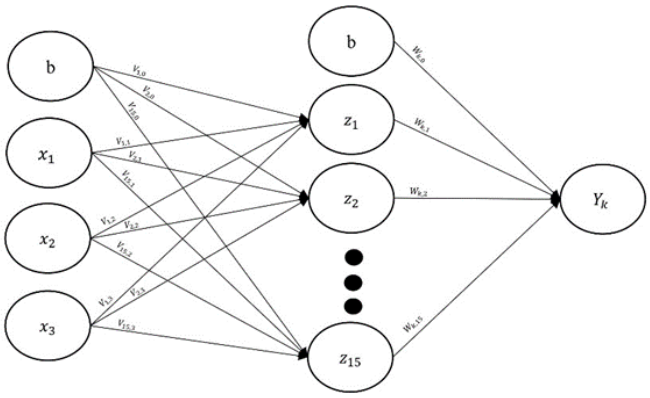
Gambar 5. Plot nilai aktual vs *output d*ata *testing*

Pada Gambar 4 dan Gambar 5 menunjukkan bahwa grafik antara data aktual dengan output jaringan pada data training dan data testing saling berdekatan. Dengan demikian, dapat dikatakan bahwa model BPNN yang terbentuk sudah sesuai dan dapat digunakan sebagai model untuk prediksi tingkat kemiskinan di DIY.



Gambar 6. Plot nilai aktual vs *output* data *training* dan *testing*

Pada Gambar 6 menunjukkan plot perbandingan nilai aktual dan *output* jaringan pada data *training* dan data *testing* terlihat sangat berdekatan. Hal ini memperkuat bahwa model BPNN yang terbentuk cocok digunakan sebagai model prediksi tingkat kemiskinan di DIY. Arsitektur BPNN dengan 15 neuron *hidden layer* dan *input* x_1, x_2, dan x_3 untuk prediksi tingkat kemiskinan di DIY adalah sebagai berikut.



Gambar 7. Model prediksi tingkat kemiskinan

3. Prediksi Tingkat Kemiskinan

Proses prediksi ini menggunakan arsitektur jaringan terbaik yang diperoleh dari pengujian data pada data training dan data testing yaitu dengan x_1, x_2, dan x_3 sebagai *input* dan 15 neuron pada *hidden layer*. Pada prediksi data targetnya adalah semua data yaitu data *training* dan data *testing*. Nilai *input* untuk prediksi tahun 2022 adalah data tahun 2021 yaitu *0,128*, sebelum dilakukan pembelajaran data harus dinormalisasi menjadi *-1.11078514308116*.

BPNN merupakan jaringan *multilayer* dengan lapisan pertama yang berupa *input layer*, lapisan kedua *hidden layer*, dan lapisan ketiga *output layer*. *Output layer* merupakan hasil prediksi model BPNN dengan rumus sebagai berikut.

$$y\_net_k = w_{k0} + \sum_{j=1}^{15} w_{kj}.z_j$$

$$y_k = y\_net_k$$

Operasi keluaran lapisan *input* ke – j ke lapisan tersembunyi

$$z\_net_j = v_{j0} + \sum_{i=1}^{3} x_i v_{ji}$$

$$= \begin{bmatrix} -3.6454 \\ \vdots \\ -3.5575 \end{bmatrix} + \sum_{i=1}^{3}(-1.1108). \begin{bmatrix} 1.7698 & 0.3541 & 2.0804 \\ \vdots & \vdots & \vdots \\ -1.6617 & -1.32 & 1.4548 \end{bmatrix}$$

diperoleh

$$z_j = f\left(z\_net_j\right) = \frac{1-e^{-z\_net_j}}{1+e^{-z\_net_j}}$$

$$= f\left(\begin{bmatrix} -3.6454 \\ \vdots \\ -3.5575 \end{bmatrix} + \sum_{i=1}^{3}(-1.1108). \begin{bmatrix} 1.7698 & 0.3541 & 2.0804 \\ \vdots & \vdots & \vdots \\ -1.6617 & -1.32 & 1.4548 \end{bmatrix}\right)$$

$$= \frac{1-e^{-\left(\begin{bmatrix} -3.6454 \\ \vdots \\ -3.5575 \end{bmatrix} + \sum_{i=1}^{3}(-1.1108). \begin{bmatrix} 1.7698 & 0.3541 & 2.0804 \\ \vdots & \vdots & \vdots \\ -1.6617 & -1.32 & 1.4548 \end{bmatrix}\right)}}{1+e^{-\left(\begin{bmatrix} -3.6454 \\ \vdots \\ -3.5575 \end{bmatrix} + \sum_{i=1}^{3}(-1.1108). \begin{bmatrix} 1.7698 & 0.3541 & 2.0804 \\ \vdots & \vdots & \vdots \\ -1.6617 & -1.32 & 1.4548 \end{bmatrix}\right)}}$$

$$z_j = \begin{bmatrix} -0.999510665365654 \\ \vdots \\ -0.730950186023352 \end{bmatrix}$$

Operasi keluaran pada lapisan tersembunyi dengan neuron tambahan menuju ke lapisan *output* adalah

$$y\_net_k = w_{k0} + \sum_{j=1}^{15} w_{kj} \, . \, z_j$$

$$= (-0.3882) + \sum_{j=1}^{15} [-0.5626 \quad \dots \quad -0.0347] . \begin{bmatrix} -0.9995 \\ \vdots \\ -0.731 \end{bmatrix}$$

sehingga

$$y_k = y\_net_k$$

$$= (-0.3882) + \sum_{j=1}^{15} [-0.5626 \quad \dots \quad -0.0347] . \begin{bmatrix} -0.9995 \\ \vdots \\ -0.731 \end{bmatrix}$$

$$y_k = 0.456302338519485$$

diperoleh nilai y_k*=0,4563* yang kemudian didenormalisasikan menggunakan fungsi *poststd*, hasil prediksi tingkat kemiskinan untuk tahun berikutnya atau tahun 2022 adalah sebesar *0,1502*. Untuk memprediksi tingkat kemiskinan di tahun 2023 dilakukan dengan langkah yang sama seperti awal dengan menambahkan data tingkat kemiskinan tahun 2021 sebagai *input* nya dan hasil prediksi tingkat kemiskinan tahun 2022 sebagai targetnya.

4. Perhitungan tingkat akurasi dan *MAPE*

Berikut merupakan perhitungan manual dengan menggunakan rumus *MAPE*.

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{x_t - \hat{x}_t}{x_t} \right| \times 100\%$$

$$MAPE = \frac{1}{11} \left(\left| \frac{0.1614 - 0.1614103}{0.1614} \right| + \left| \frac{0.1588 - 0.1587398}{0.1588} \right| + \cdots + \left| \frac{0.128 - 0.1259899}{0.128} \right| \right) \times 100\%$$

$$MAPE = \frac{1}{11} (0.00638 + 0.03792 + \cdots + 1.570389) \times 100\%$$

$$MAPE = \frac{1}{11} (5.09309959) \times 100\%$$

$$MAPE = 0.463009054$$

$$MAPE \approx 0.463\%$$

Akurasi = $100\% - MAPE$

Akurasi = $100\% - 0.463\%$

Akurasi = 99.537%

SIMPULAN

1. Model algoritma JST *Backpropagation* untuk prediksi tingkat kemiskinan terbaik adalah jaringan dengan 15 neuron pada *hidden layer*. Hal ini dikarenakan arsitektur ini menghasilkan nilai *MAPE* terkecil pada data *training* sebesar 0,02414988 dan data *testing* sebesar 1,633300171. Dalam penelitian ini model BPNN yang dibangun melalui arsitektur

jaringan dengan 15 neuron pada *hidden layer* dengan *input* $x_1, x_2, dan\ x_3$ dapat digunakan sebagai model prediksi tingkat kemiskinan di DIY. Gambar 8 menunjukan plot perbandingan nilai aktual dan *output* jaringan pada data *training* dan data *testing* terlihat sangat berdekatan. Hal ini memperkuat bahwa model BPNN yang terbentuk cocok digunakan sebagai model prediksi tingkat kemiskinan di DIY.

2. Tingkat akurasi *backpropagation* untuk memprediksi tingkat kemiskinan di DIY dengan menggunakan model terbaik yaitu jaringan 15 neuron pada *hidden layer* mendapatkan tingkat akurasi yang cukup tinggi. Hasil prediksi tingkat kemiskinan untuk tahun berikutnya atau tahun 2022 adalah sebesar 0,1502 dengan tingkat akurasi sebesar 99,537% dan *MAPE* sebesar 0,463%.

DAFTAR PUSTAKA

Aprizal, Y., Zainal, R. I., & Afriyudi, A. (2019). Perbandingan Metode Backpropagation dan Learning Vector Quantization (LVQ) Dalam Menggali Potensi Mahasiswa Baru di STMIK PalComTech. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *18*(2), 294–301. https://doi.org/10.30812/matrik.v18i2.387

Azid, I. A., Yusoff, A. R., Seetharamu, K. N., & Ahmad, A. L. (2017). *Application of Back Propagation Neural Network in Predicting Palm Oil Mill Emission. ASEAN Journal on Science and Technology for Development*, 20(1), 71–86. https://doi.org/10.29037/ajstd.376

Badan Pusat Statistik. (2017). *Badan Pusat Statistik*(pp.335–358). https://doi.org/10.1055/s-2008-1040325

Data, C., Rotan, E., & Sintetis, P. (2014). *CLUSTERING DATA EKSPOR ROTAN PLASTIK SINTETIS.*

Hijriah, N., & Narang, Z. (2020). *Prediction of Fleet Demand Needs Using Backpropagation Artificial Neural Networks and Fuzzy Time Series in Sea Release Transport System.* 9(12), 323–326.

Khusniyah, T. W., & Sutikno, S. (2016). *Prediksi Nilai Tukar Petani Menggunakan Jaringan Syaraf Tiruan Backpropagation. Scientific Journal of Informatics*, 3(1), 11–18. https://doi.org/10.15294/sji.v3i1.4970