Forecasting the tuberculosis morbidity rate in Indonesia using temporal convolutional neural network and exponential smoothing

A A A W Kusuma, S Mardiyati*, and D Lestari
Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok 16424, Indonesia

*sri_math@sci.ui.ac.id

Abstract. Tuberculosis (TB) morbidity rate in Indonesia shows the number of population in Indonesia who suffer from TB. The TB morbidity rate can be used by insurance companies to predict a person's risk of TB so that insurance companies can determine the premiums that will be charged to insurance applicants based on the risks. Thus, the ability to estimate the TB morbidity rate accurately is essential for insurance companies to be able to determine the right premium amount while remaining competitive. This study compared two models that can be used to predict TB morbidity rate in Indonesia. The model was built using the temporal convolutional neural network (TCNN) and exponential smoothing methods. The data analyzed in this study are data obtained from the official website of the Ministry of Health of the Republic of Indonesia. Before the model was built, the data used in this study were compiled into training and validation datasets. The model is built using a training dataset and validated using the validation dataset. The results of the model's validation are then evaluated and compared based on the value of the mean squared error (MSE). The result of this study shows that the TCNN model provides lower MSE compared to exponential smoothing.

1. Introduction
According to WHO, one of the 10 highest causes of death in the world is tuberculosis (TB), where Indonesia is in 2nd rank in terms of the number of TB sufferers [1]. To help alleviate the high cost of TB treatment, residents can join health insurance that covers TB disease treatment. Some premium must be paid by residents who join health insurance that covers TB disease treatment. Insurance company must calculate the right amount of premium but remain competitive. TB morbidity rates is one of the component that can be used to calculate the premium. The TB morbidity rate can indicate the risk of TB occurrence in the population of a region. Accurate prediction of the TB morbidity rate can help the insurer to calculate the right amount of premium.

The morbidity rate data is in the form of time series data, therefore the models developed to predict time series data can be used to predict TB morbidity rate. In 2015, ARIMA and ARIMA-ARCH models are used to predict the morbidity rate of TB disease in Xinjiang Province, China. [2]. Apart from ARIMA, Recurrent Neural Network (RNN) can also be used to predict time series data, such as predicting stock prices [3]. RNN and LSTM models also used to predict TB morbidity rate in Xinjiang Province, China [4]. In this study, the results showed that the RNN model combined with LSTM gave a lower residual mean squared error than the ARIMA-ARCH model.
Several machine learning models have been developed so that they can be used to forecast time series data. One of them is the development of the Convolutional Neural Network (CNN). The development from CNN has been used to forecast the S&P500 financial time series [5]. In addition, CNN development is also used to model several series data, such as MNIST and P-MNIST sequences, copy memory, and addition problems [6]. In this study, it was found that the development of CNN for series data gave better results than RNN and LSTM.

In addition to these methods, there is one time series data forecasting model that has been widely used in forecasting, namely exponential smoothing. This model can be used to forecast long and short time series [7]. Exponential smoothing has been applied to complete forecasting in various fields including predicting global radiation [8] and short-term electricity demand [9].

In this study, the TB morbidity rate in Indonesia was predicted using the development of CNN, namely Temporal Convolutional Neural Network (TCNN) and Exponential Smoothing model. Forecasting results from the two models are then compared based on the value of the Mean Squared Error (MSE).

2. Method

2.1 Temporal Convolutional Neural Network (TCNN)

Convolutional Neural Network (CNN) can be used for forecasting time series data problems [5, 6, 10]. Standard CNNs are designed for image processing but after being modified with a one-dimensional filter, they can be used to predict time series data [10]. In the next section, the CNN development for time series data will be referred as TCNN. TCNN uses causal convolution. Suppose there is a time series data \( x(0), x(1), ..., x(t - 1) \), then to predict \( x(t) \) only values from \( x(t - 1), x(t - 2), ..., x(0) \) will be used. TCNN also uses 1D Fully-Convolutional Network (FCN) architecture [11]. The TCNN model also uses a dilated convolution [5]. Suppose that a filter \( w = [w(0), w(1), ..., w(M - 1)] \) is given and \( x = [x(0), x(1), ..., x(N - 1)] \) input, then the convolutional operation of \( w \) and \( x \) is defined as [12]:

\[
(w * x)(i) = \sum_{j=0}^{M-1} w(j)x(i - j)
\]  

(1)

whereas, the dilated convolution operation of \( w \) and \( x \) is defined as:

\[
(w *_{d} x)(i) = \sum_{j=0}^{M-1} w(j)x(i - dj)
\]  

(2)

with \( i = 0,1, ..., t - 1 \). The TCNN model in this paper also implements a residual connection and a skip connection [13].

The TCNN model training process involves two main processes, namely forward pass and backpropagation. In the forward pass process, the input vector is entered and processed at each network layer until the output of the model is obtained. Meanwhile, in the backpropagation process, the error from the output is propagated back to the first layer to update the weights so that a combination of weights can produce a minimum error. Both processes are repeated 200 times iterations or repetitions.

In this study, the length of the input vector is determined by the size of the receptive field used. Receptive field calculations can be done using equations:

\[ r = 2^{D-1} \times k \]

(3)

with \( r \), \( D \), and \( k \) shows the size of the receptive field, the number of dilation factors used, and the size of the filter, consecutively. In this study, the filter size, \( k \), is 2, with dilation factor are 1, 1 and 2, or 1, 2, and 4. Therefore, the sizes of the receptive fields used are 2, 4, and 8.

Forward pass steps:

1. Prepare input vectors and initiate weight and bias values for each layer. The input size can be calculated with an equation \( r = 2^{D-1} \times k \), with \( D \) the number of dilation factors, and \( k \) is the filter size. Suppose that the input vector \( x(0) = [x(0), x(1), ..., x(t - 1)] \) is given, with \( t \) in the same size as the receptive field. This input will be used to predict the value of \( x(t) \).

2. Perform convolutional operations using equations:

\[ x^{l}(i, h) = (w^{l}_{h} \ast x)(i, h) = w^{l}_{h}x(i) + b^{l}_{h} \]

(4)
with \( h = 1, 2, \ldots, H \) where \( H \) is the number of filters and \( i = 0, 1, \ldots, t - 1 \), so that it is obtained \( x^1 = [x^1(i, h)] \).

3. The following operations occur on the residual block. Perform convolutional operations using equations:

\[
x^2(i, h) = (w^2_h \ast_d x^1)(i, h) = \sum_{j=0}^{1} \sum_{m=1}^{M^1} (w^2_h(j, m)x^1(i - jd, m) + b^2_h
\]

with \( M^1 \) is the number of channels, that is the number of filters used in the previous convolution layer from the residual block input and \( d = 1 \) is the dilation factor. With \( h = 1, 2, \ldots, H \) where \( H \) is the number of the filters, and \( i = 0, 1, \ldots, t - 1 \), so that is obtained \( x^2 = [x^2(i, h)] \) with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \).

4. Transform the element on \( x^2 \) using ReLU function:

\[
y^1(i, h) = \max \left(0, x^2(i, h)\right) = \begin{cases} x^2(i, h) & \text{if} \ x^2(i, h) \geq 0 \\ 0 & \text{if} \ x^2(i, h) < 0 \end{cases}
\]

so that is obtained \( y^1 = [y^1(i, h)] \), with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \).

5. Repeat steps 3 and 4 to obtain \( x^3 = [x^3(i, h)] \) and \( y^2 = [y^2(i, h)] \) with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \). Define \( s^1 = y^2 = [y^2(i, h)] \) as the value that will be used in the skip connection process.

6. Perform convolutional operations using equations:

\[
x^4(i, h) = (w^4_h \ast x^3)(i, h) = \sum_{m=1}^{M^1} w^4_h(m)x(i) + b^4_h
\]

with \( M^1 \) is the number of channels, which is the number of filters used in the previous convolution layer, \( h = 1, 2, \ldots, H \) with \( H \) is the number of filters and \( i = 0, 1, \ldots, t - 1 \), so that is obtained \( x^4 = [x^4(i, h)] \).

7. Add each of the corresponding elements of \( x^4 = [x^4(i, h)] \) and \( y^2 = [y^2(i, h)] \), so that is obtained:

\[
\mathbf{rc}^1 = [r^1(i, h)] = [x^4(i, h)] + [y^2(i, h)]
\]

8. Transform the elements on \( x^5 \) using Tanh function:

\[
r^1(i, h) = \tanh (\mathbf{rc}^1(i, h)) = \frac{e^{\mathbf{rc}^1(i, h)} - e^{-\mathbf{rc}^1(i, h)}}{e^{\mathbf{rc}^1(i, h)} + e^{-\mathbf{rc}^1(i, h)}}
\]

so that is obtained \( r^1 = [r^1(i, h)] \) with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \).

9. If dilation values 1 and 2 are used, the process is continued by repeating steps (3) to (5) with a value of \( d = 2 \) using \( r^1 = [r^1(i, h)] \) as input for step (3), so that is obtained \( x^5, x^6, y^3, y^4, \) and \( s^2 \).

10. If three dilation values are used, namely 1, 2 and 4, the process is continued by repeating steps (3) to (7) with values \( d = 2 \) using \( r^1 = [r^1(i, h)] \) as input for step (3), so that is obtained \( x^5, x^6, x^7, \mathbf{rc}^2, y^3, y^4, s^2, \) and \( r^2 \). After that, continue by repeating steps (3) to (5) with values \( d = 4 \) using \( r^2 = [r^2(i, h)] \) as input for step (3), so that is obtained \( x^8, x^9, y^5, y^6, \) and \( s^3 \).

11. If using one dilation value, then take the last value from each channel \( r^1 \), that is \( \mathbf{sc}^\text{out} = [r^1(t - 1, h)] \) with \( h = 1, 2, \ldots, H \).

12. If two dilation values are used, namely 1 and 2, then add the corresponding element of \( s^1 = [y^2(i, h)] \), \( s^2 = [y^4(i, h)] \) so that is obtained:

\[
\mathbf{sc} = [\mathbf{sc}(i, h)] = [y^2(i, h)] + [y^4(i, h)]
\]

with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \). Then take the last element of each channel on \( \mathbf{sc} \), that is \( \mathbf{sc}^\text{out} = [\mathbf{sc}(t - 1, h)] \), with \( h = 1, 2, \ldots, H \).

13. If three dilation values are used, namely 1, 2 and 4, then add the corresponding elements of \( s^1 = [y^2(i, h)] \), \( s^2 = [y^4(i, h)] \), and \( s^3 = [y^6(i, h)] \), so that is obtained:

\[
\mathbf{sc} = [\mathbf{sc}(i, h)] = [y^2(i, h)] + [y^4(i, h)] + [y^6(i, h)]
\]

with \( i = 0, 1, \ldots, t - 1 \) and \( h = 1, 2, \ldots, H \). Then take the last element of each channel on \( \mathbf{sc} \), that is \( \mathbf{sc}^\text{out} = [\mathbf{sc}(t - 1, h)] \), with \( h = 1, 2, \ldots, H \).

14. Perform the following operation to obtain the model output that will be used as the forecast value.
\[ \hat{x}(t) = \sum_{i=0}^{H-1} (s^{\text{out}}(i)w^{\text{out}}(i)) + b^{\text{out}} \]  

(12)

with \( H \) is the number of filters.

Backpropagation steps:

The weights and biases of the model are updated to find the combination of weights that will produce the minimum error. The updating of the weights is done using the stochastic gradient descent [14] method, where the weights are updated using the equation:

\[ w_h^{l}(t + 1) = w_h^{l}(t) - \eta \nabla E(w_h^{l}(t)), \quad \tau = 1, 2, ..., T \]  

with \( \nabla E(w_h^{l}(\tau)) \) the gradient vector of the change in error value against weight, \( l \) is the layer index of the model architecture, \( h \) represents the filter index, \( \eta \) is a constant of the learning rate, as well \( T \) is the number of iterations or epochs. In this study, the number of iterations is 200.

1. Calculate the error from the model’s prediction using the equation:

\[ MSE = \frac{1}{N} \sum_{t=0}^{N-1} (x(t) - \hat{x}(t))^2 \]  

(14)

with \( N \) is the amount of data, \( x(t) \) is the actual value, and \( \hat{x}(t) \) is the forecast value generated by the model.

2. Calculate the derivative of the error function (MSE) for each weight in the model. For example, the gradient vector of \( w^{\text{out}}(i) \) in step 14 of the forward pass is calculated, then that is obtained.

\[ \frac{\partial MSE}{\partial w^{\text{out}}(i)} = \frac{\partial}{\partial w^{\text{out}}(i)} \sum_{t=0}^{N-1} (x(t) - \hat{x}(t))^2 \]  

(15)

with \( i = 0, 1, ..., H - 1 \) where \( H \) is the number of filters, and \( N \) is the number of data.

3. Update the weights using the following equation.

\[ w^{\text{out}}(i)_{\text{baru}} = w^{\text{out}}(i) - \eta \frac{\partial MSE}{\partial w^{\text{out}}(i)} \]  

(16)

with \( i = 0, 1, ..., H - 1 \) where \( H \) is the amount of filters, and \( \eta \) is the learning rate constant.

4. Repeat steps (2) and (3) for all weights and biases in the model.

2.2 Exponential Smoothing

In this study, we also analyzed the data with the Simple Exponential Smoothing (SES) model. The SES model was chosen because there is no trend or seasonal pattern in the data. The forecast value of the SES model is calculated using the following equation [15]:

\[ \hat{x}(t + h) = \hat{x}(t) + \lambda x(t) + (1 - \lambda)\hat{x}(t - 1) \]  

(17)

with \( t = 0, 1, ..., j \), where \( j \) is the amount of data series used in the smoothing process, \( h = 1, 2, ..., \), then \( \hat{x}(t + 1) \) is the forecasting value at time \( t + 1 \), \( \hat{x}(t) \) is an exponentially weighted average at time \( t \). In this research \( \hat{x}(-1) = x(0) \) and the value of \( \lambda \) is selected from 0.1, 0.2, 0.3, ..., 0.9.

3. Result and Discussion

3.1 Data

The data analyzed in this research are the total population of Indonesia (JP) and the number of Indonesians with health complaints (JPPK) for TB disease from 2004 to 2018, obtained from the official website of the Ministry of Health of the Republic of Indonesia. We calculate the value of TB morbidity rate in Indonesia using equations [16]:

\[ TM = \left(\frac{JPKK}{JP}\right) \times 100 \]  

(18)

with TM shows the value of TB morbidity rates in Indonesia. In this paper, the value of the morbidity level will then be symbolized by \( y(t) \). After calculating the value of the morbidity rate using the above equation, then standardizing the value of the morbidity rate is carried out using the following equation.
\[
    z(t) = \frac{y(t) - \bar{y}}{s}; \quad t = 1, 2, ..., n
\]
with \(\bar{y} = 0.0216\) and \(s = 0.1318\) is the mean and standard deviation of the morbidity rate \(y(t)\). The results of that calculation are presented in Table 1 below.

**Table 1.** TB morbidity rate in Indonesia from 2004 to 2018

| \(t\) | year | Morbidity rate \((y(t))\) | \(z(t)\) |
|-------|------|--------------------------|---------|
| 0     | 2004 | 0.0989                   | -1.5756 |
| 1     | 2005 | 0.1188                   | -0.6257 |
| 2     | 2006 | 0.1249                   | -0.3318 |
| 3     | 2007 | 0.1188                   | -0.6251 |
| 4     | 2008 | 0.1305                   | -0.0637 |
| 5     | 2009 | 0.1274                   | -0.2147 |
| 6     | 2010 | 0.1274                   | -0.2118 |
| 7     | 2011 | 0.1332                   | 0.0639  |
| 8     | 2012 | 0.1324                   | 0.0248  |
| 9     | 2013 | 0.1317                   | -0.0102 |
| 10    | 2014 | 0.1131                   | -0.8949 |
| 11    | 2015 | 0.1295                   | -0.1121 |
| 12    | 2016 | 0.1360                   | 0.1977  |
| 13    | 2017 | 0.1623                   | 1.4534  |
| 14    | 2018 | 0.1932                   | 2.9258  |

**3.2 Input and Target Vector for TCNN**

In the TCNN model, before being used for model training, data is prepared and transform into a vector that have length as long as the size of the receptive field and one observation after that as the target. For \(x(t)\) with \(t = 0, 1, ..., 14\) the input vector are prepared as follow.

\[
    x(0) = [x(0), x(1), ..., x(r − 1)] \quad \text{is used to acquire} \quad \hat{x}(r) \\
    x(1) = [x(1), x(2), ..., x(r)] \quad \text{is used to acquire} \quad \hat{x}(r + 1) \\
    \text{...} \\
    x(j) = [x(j), x(j + 1), ..., x(j + r − 1)] \quad \text{is used to acquire} \quad \hat{x}(14)
\]

where \(r\) is the size of the receptive field, \(j\) is the number of input vectors formed which can be calculated by \(14 − r\), \(x\) is the morbidity rate value that has been standardized or not and is presented in Table 1, and \(\hat{x}\) is the forecasting value at time \(t\).

**3.3 Error of Forecasting Results from TCNN and Exponential Smoothing Models with 2 Validation Data**

The error of forecasting results in this study is calculated using Mean Squared Error (MSE). The MSE value is calculated both on the data used in modeling (training data) and data validation. The TCNN model in this section is built using input and target vectors apart from the two vectors and the last target formed from the previous vector formation, while in exponential smoothing 13 first observation in the data are used in the smoothing process without the need to divide into several previous vectors. Forecasting results from a trained model using standardized data are transformed first using the equation:

\[
    y(i) = sz(i) + \bar{y}; \quad i = 1, 2, ..., n
\]

with \(\bar{y} = 0.0216\) and \(s = 0.1318\). The transformed value is then used in the MSE calculation.

The TCNN model and exponential smoothing with the lowest training and validation MSE value then compared to see the performance of the forecast results. The model with the lowest MSE is presented in Table 2.
Table 2. Comparison of the lowest MSE value for the TCNN and ES models with 2 validation data

| Model | Training MSE | Validation MSE |
|-------|--------------|----------------|
| TCNN trained on standardized data, receptive field 8, and 8 filters | $2.19 \times 10^{-6}$ | $1.16 \times 10^{-4}$ |
| ES ($\lambda = 0.8$) | $9.69 \times 10^{-5}$ | $2.13 \times 10^{-3}$ |
| ES ($\lambda = 0.9$) | $9.80 \times 10^{-5}$ | $2.04 \times 10^{-3}$ |

Based on Table 2, it can be seen that the TCNN model with a receptive field size of 8 and 8 filters that is trained on standardized data gives lower MSE values when compared to the ES model with $\lambda = 0.8$ and $\lambda = 0.9$. It means, with 2 validation data, the TCNN model gives better forecasting result than ES models. The graph of the prediction results of the three models can be presented in Figure 1 below.

![Graph comparison of forecasting results with 2 validation data: (a) ES with $\lambda = 0.8$, (b) ES with $\lambda = 0.9$, and (c) TCNN](image)

3.4 Error of Forecasting Results from TCNN and Exponential Smoothing Models with 1 Validation Data

The TCNN model in this section is built using input and target vectors apart from one vector and the last target formed from the previous vector formation. For exponential smoothing models, 14 first observation of the data are used in the smoothing process. The results of forecasting from a trained model using standardized data are transformed first using the equation 20:

The TCNN model and exponential smoothing with the lowest MSE value of training and validation were then compared to see the performance of the forecast results. The model with the lowest MSE is presented in Table 3.
Table 3. Comparison of the lowest MSE value for TCNN and ES models with 1 validation data

| Model                                           | Training MSE | Validation MSE |
|------------------------------------------------|--------------|----------------|
| TCNN trained on non-standardized data receptive field 2, and 8 filters | $1.75 \times 10^{-4}$ | $1.76 \times 10^{-5}$ |
| TCNN trained on standardized data receptive field 8, and 8 filters | $9.70 \times 10^{-7}$ | $2.40 \times 10^{-5}$ |
| ES ($\lambda = 0.9$)                              | $1.43 \times 10^{-5}$ | $1.13 \times 10^{-3}$ |

Based on Table 3, it can be seen that with 1 validation data, the TCNN model with a receptive field size 2 and 8 filters trained on non-standardized data produces a lower validation MSE value than the two other models in Table 3, but that model has higher training MSE. It can be presumed that the model is underfit or it cannot find the pattern of the data. Then with 1 validation data, the TCNN model with a receptive field size of 8 and 8 filters which is trained using standardized data provides a lower MSE training value when compared to the other models in Table 3. The model also has a low validation MSE. To make it clearer, the graphs of the prediction results from the three models can be presented in Figure 2 below.

![Comparison graph of forecasting results with 1 validation data](image)

Based on Figure 2 it can be seen that the TCNN model with a receptive field size of 8 and 8 filters trained on standardized data gives the best forecasting results than the other models.

4. Conclusion

The performance of the TCNN and ES models was compared based on the MSE value of training and validation. If the model is evaluated using two validation data, the TCNN model with a receptive field size of 8 and 8 filters trained with standardized data gives lower training and validation MSE values when compared to the ES model with $\lambda = 0.8$ and $\lambda = 0.9$. So, for 2 validation data, TCNN model gives lower MSE than ES models. For one validation data, the TCNN model with a receptive field size of 2 and 8 filters trained on non-standardized data produced the lowest validation MSE value than the other models but there is underfitting problem in the model. Then the TCNN model with a receptive field size
of 8 and 8 filters which is trained using standardized data gives the lowest MSE training value when compared to the other models. From the graph in Figure 2, it can be seen that TCNN model with a receptive field size of 8 and 8 filters which is trained using standardized data gives better forecasting results than the other models. From those results, we conclude that the TCNN model provides lower MSE compared to ES.

Acknowledgments
The authors would like to thank the DRPM UI for fully funded this research.

References
[1] World Health Organisation 2018 Global Health TB Report
[2] Zheng Y L, Zhang L P, Zhang X L, Wang K and Zheng Y J 2015 Forecast model analysis for the morbidity of tuberculosis in Xinjiang, China PLoS One 10 1–13
[3] Berradi Z and Lazaar M 2019 Integration of principal component analysis and recurrent neural network to forecast the stock price of casablanca stock exchange Procedia Comput. Sci. 55–61
[4] Singh J, Dhyani T, Jain A and Agrawal P 2018 Using Deep Learning for Forecasting Tuberculosis Morbidity Rate ICICSS 2018
[5] Borovykh A, Bohte S and Oosterlee C W 2017 Conditional time series forecasting with convolutional neural networks Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol 10614 LNCS pp 729–30
[6] Bai S, Koller J Z and Koltun V 2018 An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling
[7] Kirshners A and Borisov A 2013 A Comparative Analysis of Short Time Series Processing Methods Inf. Technol. Manag. Sci. 15 65–9
[8] Yang D, Sharma V, Ye Z, Lim L I, Zhao L and Aryaputra A W 2015 Forecasting of global horizontal irradiance by exponential smoothing, using decompositions Energy 81 111–9
[9] Rendon-Sanchez J F and de Menezes L M 2019 Structural combination of seasonal exponential smoothing forecasts applied to load forecasting Eur. J. Oper. Res. 275 916–24
[10] Geng Y, Su L, Jia Y and Han C 2019 Seismic Events Prediction Using Deep Temporal Convolution Networks J. Electr. Comput. Eng. 2019
[11] Shelhamer E, Long J and Darrell T 2017 Fully Convolutional Networks for Semantic Segmentation 2017 IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE) pp 640–51
[12] Weisstein E W 1999 Cross-Correlation MathWorld--A Wolfram Web Resour. 1–2
[13] Oord A van den, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A and Kavukcuoglu K 2016 WaveNet: A Generative Model for Raw Audio 1–15
[14] Zhang J 2019 Gradient Descent based Optimization Algorithms for Deep Learning Models Training
[15] Montgomery D C, Jennings C L and Kulahi M 2015 Introduction to time series analysis and forecasting (second edition) (New Jersey: John Wiley & Sons, Inc)
[16] BPS 2002 Angka Kesakitan/Morbiditas/Persentase Penduduk Yang Mempunyai Keluhan Kesehatan 5–6