Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

# Python script fuzzy time series Markov chain model for forecasting the number of diseases cocoa plant in Bendungan district

**Ajeng Berliana Salsabila[a], Firdaniza Firdaniza[b], Budi Nurani Ruchjana[b*] and Atje Setiawan Abdullah[c]**

*[a]Master of Mathematics Study Program, Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Sumedang 45363, Indonesia*
*[b]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Sumedang 45363, Indonesia*
*[c]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Sumedang 45363, Indonesia*

C H R O N I C L E

*Article history*:
Received: December 2, 2022
Received in revised format: January 29, 2023
Accepted: March 9, 2023
Available online: March 9, 2023

*Keywords:*
*Python*
*Fuzzy Time Series Markov Chain*
*Cocoa plant diseases*

A B S T R A C T

Cocoa is a plantation commodity whose role is essential for the economy, so it is necessary to be aware of its health to maximize production. Cocoa plant disease data is a time series data because it is observed continuously. One of the time series forecasting models is the Fuzzy Time Series Markov Chain (FTS-MC), a combination of the Fuzzy Time Series (FTS) and Markov Chain models. The model uses the principle of fuzzy logic by transferring FTS data to fuzzy logic and using the obtained fuzzy logic groups to derive the Markov chain transition matrix. In this research, a Python script of the FTS-MC model was built in the Google Colaboratory to forecast the number of cocoa plant diseases in Bendungan District to simplify and speed up the data processing. Python was used in this research because of its easy-to-use, flexible, and open-source syntax. In building Python scripts, libraries and functions are needed by utilizing loop processes and if-else statements. Based on the processing results, forecasting with the FTS-MC model using Python only takes less than 1 minute.

## 1. Introduction

In the era of digitalization, technology is an essential part of life, one of which is in science (Hashim, 2018). The existence of technology can make things better and faster for human work (Cascio & Montealegre, 2016). Software is one of the technologies used in science and has an important role, especially in proving scientific publications (Kanewala & Bieman, 2014). Python is one of the software that is widely used in scientific research because the syntax is easy to use, open source, and flexible (Petrelli, 2021). Sayama et al. (2013) used Python to analyze and simulate complex system models with adaptive networks. Zwicke et al. (2016) created a framework using Python as a more efficient alternative to performing Jacobian computations in finite code elements with TAPENADE. Listewnik & Aftewicz (2023) analyzed a 3D rotary magnetic field scanner for research and minimizing UUV magnetic fields with a polynomial regression approach using Python. Zhu & Liu (2023) used Python software to predict the supply chain risks of prefabricated buildings. Wang et al. (2022) created a model for recognizing and classifying extract electrocardiogram (ECG) images using Python. Canlas et al. (2022) explored the perceptions of the people who have undergone vaccination regarding various side effects to provide input to vaccine manufacturers for the statistical analysis using Python in Anaconda Spider. Leon-Alcaide et al. (2020) introduced a Python package that implements an evolutionary strategy for finding prototypes of time series analysis.

* Corresponding author.
E-mail address: budi.nurani@unpad.ac.id (B. N. Ruchjana)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2023.3.009

Several platforms provide "ready-to-use" Python, one of which is Google Colaboratory or Google Colab. Google Colab is a Google product that can run Python in the browser; users can use Python's library for free and easily share (Paper, 2021). Python scripts were built on Google Colaboratory for this research's Fuzzy Time Series Markov Chain (FTS-MC) model. Tsaur (2012) developed the FTS-MC model to forecast the exchange rate between Taiwan and the US dollar using the principle of fuzzy logic developed by Zadeh (1996). The FTS-MC model combines the FTS model and the Markov chain. The FTS-MC model was chosen in this research because it is superior to many methods available in the literature in model accuracy (Alyousifi et al., 2021). Song and Chissom (1993) developed the FTS model for linguistic data type problems. Song & Chissom (1994) developed their previous research using the FTS time-variant model. Chen (1996) proposed a new method based on FTS using simplified arithmetic operations. Then, Tsaur (2012) combines the FTS model and the Markov chain by transferring the FTS data to the fuzzy logic group and using the obtained fuzzy logic group to derive the Markov chain transition matrix. Combining models in time series forecasting is usually better than single models (Alyousifi et al., 2021).

The Python software used in this study is to simplify and speed up data processing. The data used in this study is data on cocoa plant diseases in Bendungan District, which are included in the time series data because they are observed from time to time. Cocoa is one of the primary plantation commodities whose function is vital for the national economy, especially as a provider of employment, a source of income, and foreign exchange (Kemenperin, 2007). So, it is necessary to be aware of its health to maximize production. This research can help relevant agencies determine future policies, especially for eradicating diseases in cocoa plants. The evaluation of the forecasting model used in this study is the Mean Absolute Percentage Error (MAPE).

## 2. Materials

### *2.1. Python*

Python is a high-level program, which means that the Python language is easy for humans to understand, just like Maple, MATLAB, and R. Python's performance is slower than low-level programs such as C and FORTRAN. Still, the Numba package can significantly improve Python's performance to approach the speed of C and FORTRAN. Python was first developed by Guido van Rossum and released in 1991 (Petrelli, 2021).

Python is modular, which means it supports using modules and packages. Packages or libraries in Python are a collection of functions and modules that function to carry out a particular task (Petrelli, 2021). Python has a standard library with an extensive collection of built-in and portable functionality. This library supports various application-level programming assignments, from text pattern matching to network scripting. Moreover, Python can be amplified with custom libraries and a broad collection of software-supporting applications such as site development, serial port access, numeric programming, and more. For example, the NumPy extension is the more effective proportionate of the MATLAB numeric programming framework (Lutz, 2013).

The advantage of Python is that the language is easy to learn. Additionally, extensive built-in run-time checking helps detect bugs and reduces development time. Python can also be used on easily nested and heterogeneous data structures. Python programming is object-oriented programming (OOP); besides that, using Python, there is support for numerical computational efficiency, and integration of Python with C and FORTRAN is almost automatic (Langtangen, 2008).

Several platforms provide "ready-to-use" Python, one of which is Google Colaboratory or Google Colab. This Google product can run Python in a browser and is like Jupyter Notebook. Using Google Colab, users can use libraries owned by Python, access the Graphics Processing Unit (GPU) for free, and easily share data (Paper, 2021).

### *2.2. Fuzzy Time Series*

***Definition 1.*** *Suppose $Y(t)$ where $(t = 0,1,2,\dots)$ is a subset of $\mathbb{R}$ which is the universe of discourse of the fuzzy set $f_i(t)$ where $(i = 1,2,3,\dots)$ defined, and $F(t)$ is the fuzzy set $f_i(t)$ where $(i = 1,2,3,\dots)$ then, $F(t)$ is Fuzzy Time Series (FTS) on $Y(t)$ where $(t = 0,1,2,\dots)$* (Song & Chissom, 1993).

### *2.3. Fuzzy Time Series Markov Chain*

Tsaur (2012) developed the Fuzzy Time Series Markov Chain (FTS-MC) by transferring FTS data to fuzzy logic groups and using them to obtain a Markov chain transition matrix. The step to forecast FTS-MC are as follows:

Step 1. Define the universe of discourse $U$ on historical data using formula (1)

$$U = [D_{min} - D_1, D_{max} + D_2] \tag{1}$$

where $D_{min}$ is minimum data, $D_{max}$ is maximum data, and $D_1$; $D_2$ are proper positive numbers.

Step 2. Partition $U$ in class interval

To partition $U$, the number of class interval $(K)$ is determined in advance based on Sturges' rules as in (2).

$$K = 1 + 3.322\log(N) \tag{2}$$

where $N$ is the amount of data used. Then, determine the length interval $(l)$ using formula (3),

$$l = \frac{[(D_{max} + D_2) - (D_{min} - D_1)]}{K}. \tag{3}$$

So, the partition of $U$ as in (4).

$$u_n = [D_{min} - D_1(n - l), D_{min} - D_1 + nl] \tag{4}$$

where $u_n$ is the $n^{th}$ interval.

Step 3. Define the fuzzy set
In defining a fuzzy set, it is necessary to determine the membership value $u_i$ for each $A_i$ with the following conditions (5).

$$\mu(u_i) = \begin{cases} 1 & ; u_i = A_i \\ 0.5 & ; u_{i+1} = A_i \; or \; u_{i-1} = A_i \\ 0 & ; other. \end{cases} \tag{5}$$

Step 4. Historical data fuzzification
Historical data fuzzification is a step to determine the fuzzy set for each data by deciding the membership value based on the membership function. If the data obtained is included in the $u_i$ interval, then the data is fuzzified to the fuzzy set $A_i$.

Step 5. Fuzzy Logical Relationship (FLR) and Fuzzy Logical Relationship Group (FLRG)

**Definition 2.** *Suppose $F(t) = A_i$ is caused by $F(t-1) = A_j$, then FLR defined as $A_i \to A_j$* (Song & Chissom, 1993).
**Definition 3.** *Suppose an FLR is formed from state $A_i$ then the state transitions to another state such as $A_i \to A_k, A_i \to A_l, A_i \to A_m$, the FLRG that is formed $A_i \to A_k, A_l, A_m$* (Song & Chissom, 1993).

Step 6. Determine the Markov chain transition probability matrix

$$p_{ij} = \frac{M_{ij}}{M_i} \; ; i, j = 1,2,3, \dots, n \tag{6}$$

where $p_{ij}$ is probability of transition $A_i \to A_j$, $M_{ij}$ is the number of transitions $A_i \to A_j$, and $M_i$ is total number of transitions $A_i$. Then the Markov chain transition probability matrix obtained as in (7).

$$\begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0n} \\ p_{10} & p_{11} & \cdots & p_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n0} & p_{n1} & \cdots & p_{nn} \end{bmatrix} \tag{7}$$

where $p_{ij} \geq 0$ and $\sum_{j=1}^{n} p_{ij} = 1$.

Step 7. Calculate the forecast value $(F(t))$
There are three conditions to get the forecast value:
1. If the FLRG of $A_i$ is empty set $(A_i \to \emptyset)$, the formula (8) is used to determine $F(t)$,

$$F(t) = m_i \tag{8}$$

where $m_i$ is middle value of $A_i$.
2. If the FLRG of $A_i$ is one-to-one $\left(A_i \to A_j\right)$ where $p_{ij} = 1$, the formula (9) is used to determine $F(t)$,

$$F(t) = m_j \tag{9}$$

where $m_j$ is middle value of $A_j$.
3. If the FLRG of $A_i$ is one-to-many $(A_i \to A_1, A_2, A_3, \dots, A_n)$, the formula (10) is used to determine $F(t)$,

$$F(t) = m_1 p_{i1} + m_2 p_{i2} + \cdots + m_{i-1} p_{i(i-1)} + Y(t-1) p_{ii} + m_{i+1} p_{i(i+1)} + m_n p_{in} \tag{10}$$

where $m_1$ is middle value of $A_1$, etc.

Step 8. Specifies the adjustment value to repair the error
If the state $A_i$ with the time $(t-1)$ or $F(t-1) = A_i$, transitions to state $A_j$ with the time $(t)$, where $s$ is $j - 1$, the adjustment value is as in formula (11),

$$D_t = \frac{l}{2} s. \tag{11}$$

Step 9. The final forecast value ($F'(t)$)

$$F'(t) = F(t) \pm D_t. \tag{12}$$

Forecasting using the FTS-MC model, if depicted in the form of a flowchart, can be seen in Fig. **1**.

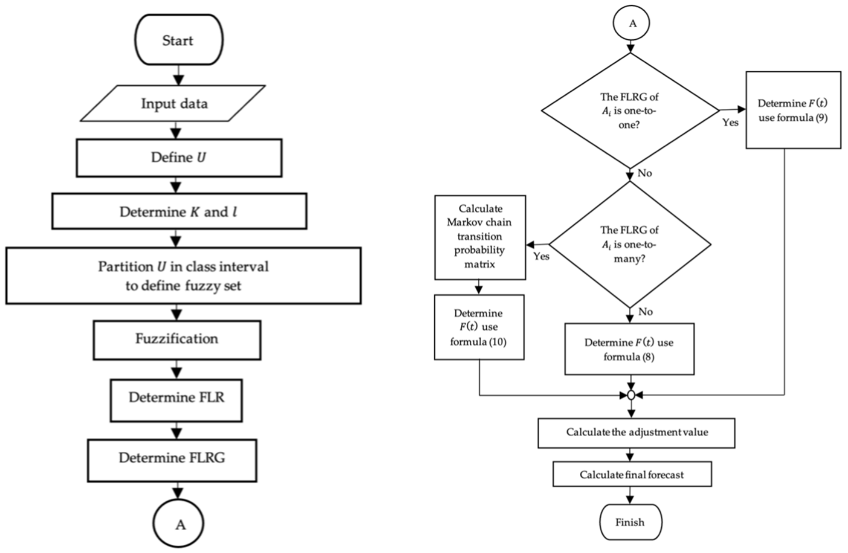

Start
Input data
Define $U$
Determine $K$ and $l$
Partition $U$ in class interval to define fuzzy set
Fuzzification
Determine FLR
Determine FLRG
A
A
The FLRG of $A_i$ is one-to-one?
Yes
Determine $F(t)$ use formula (9)
No
The FLRG of $A_i$ is one-to-many?
Yes
Calculate Markov chain transition probability matrix
Determine $F(t)$ use formula (10)
No
Determine $F(t)$ use formula (8)
Calculate the adjustment value
Calculate final forecast
Finish

**Fig. 1.** FTS-MC model procedure

## *2.4 Mean Absolute Percentage Error*

Mean Absolute Percentage Error (MAPE) evaluates a forecasting model that considers the actual values' effect. The lower the MAPE value, the more accurate the model accuracy (Lawrence et al., 2009). Formula (13) is used to calculate the MAPE value, and Table 1 is a scale to assess model accuracy based on MAPE values.

$$MAPE = \frac{\left(\sum_{t=1}^{N} \frac{|Y(t) - F'(t)|}{Y(t)}\right)}{N} \times 100\%. \tag{13}$$

**Table 1**
MAPE accuracy

| MAPE | Forecast Accuracy |
|---|---|
| **≤ 10%** | Highly accurate |
| **10% < MAPE ≤ 20%** | Good forecast |
| **20% < MAPE ≤ 50%** | Reasonable forecast |
| **50% <** | Inaccurate forecast |

## 3. Methodology

The research method used is building Python scripts on Google Colab for the FTS-MC model to forecast the number of diseases that attack cocoa plants in Bendungan District in August-October 2022. The data used in this study were obtained from the official website of PT. Center for Plantation Plant Seeding and Protection (BBPPTP) Surabaya https://simopt-bbpptp-surabaya.net from January 2016 to July 2022. Table 2 is the general procedure of this research.

**Table 2**
The general procedure of this research

| The General Procedure | |
|---|---|
| Step 1 | Open Google, then open the website https://colab.research.google.com/. |
| Step 2 | Click "New notebook" to start the data processing. |
| Step 3 | Install the library with the command "pip install library name". |
| Step 4 | After successfully installing the library, click "+Code" to create a new console, then import the library. |
| Step 5 | Building a Python Script on Google Colab for the FTS-MC model (Algorithm 1-Algorithm 12). |
| Step 6 | Input data in the form of Comma Separated Values (.csv). |
| Step 7 | Implementing Python Scripts on Google Colab for the FTS-MC models in forecasting cocoa plant diseases in Bendungan District (run program). |

## *3.1 Discussion and Result*

The process for building a Python script of the FTS-MC model on the Google Colab is as follows:

1. The universe of discourse ($U$)

Determine the maximum and minimum data (Algorithm 1).

**Algorithm 1: The universe of discourse**

```
def get_data_max_min(df):
    data_maximum = max(df[flag_file])
    data_minimum = min(df[flag_file])
    return data_maximum, data_minimum
def get_interval(data_maximum,data_minimum,adjuster):
    interval_high = round(data_maximum/adjuster)*adjuster
    if interval_high < data_maximum:interval_high += adjuster
    interval_low = round(data_minimum/adjuster)*adjuster
    if interval_low > data_minimum:interval_low -= adjuster
    return interval_high,interval_low
```

From Algorithm 1, using the "max" and "min" functions, the maximum data is 439, and the minimum is 60. The Adjuster function was built to round the maximum and minimum data. So that the values of $D_1$ and $D_2$ do not need to be determined. Then the universe of discourse is $U = [60,440]$.

2. Class interval and length interval

**Algorithm 2 : Class interval and length interval**

```
K = (1 + (3.322*math.log(data_length,10)))
K_rounded_up = round(K/1)*1
if (K_rounded_up < K): K_rounded_up += 1
def get_interval_class(interval_high,interval_low,K_rounded_up):
    return (interval_high-interval_low)/K_rounded_up
```

From Algorithm 2, the "round" function is used to round the K value, so we get $K = 8$ and $l = 47$.

3. The fuzzy set

Determine the fuzzy set interval and middle value first to get the fuzzy set.

**Algorithm 3: The fuzzy set interval and middle value**

```
def generate_interval(interval_class,K_rounded_up,interval):
    int_K_rounded_up = int(K_rounded_up)+1
    mtx_interval = [ [ 0 for i in range(2) ] for j in range (int_K_rounded_up) ]
    counter = interval[1]
    for i in range (int_K_rounded_up):
        for j in range (2):
            if (j == 0): mtx_interval[i][j] = int(counter)
            else:counter += interval_class; mtx_interval[i][j] = int(counter)
    return (mtx_interval)
```

In Algorithm 3, we get the fuzzy set interval, the middle value in Table 3 and the fuzzy set in Eq. (14).

**Table 3**
The fuzzy set interval and middle value of interval.

| $u_n$ | Interval | $m_n$ |
|---|---|---|
| 0 | [60,107] | 83 |
| 1 | (107,155] | 131 |
| 2 | (155,202] | 178 |
| 3 | (202,250] | 226 |
| 4 | (250,297] | 273 |
| 5 | (297,345] | 321 |
| 6 | (345,392] | 368 |
| 7 | (392,440] | 416 |

$$\begin{aligned} A_0 &= \left\{\frac{1}{u_0},\frac{0.5}{u_1},\frac{0}{u_2},\frac{0}{u_3},\frac{0}{u_4},\frac{0}{u_5},\frac{0}{u_6},\frac{0}{u_7}\right\} \\ A_1 &= \left\{\frac{0.5}{u_0},\frac{1}{u_1},\frac{0.5}{u_2},\frac{0}{u_3},\frac{0}{u_4},\frac{0}{u_5},\frac{0}{u_6},\frac{0}{u_7}\right\} \\ A_2 &= \left\{\frac{0}{u_0},\frac{0.5}{u_1},\frac{1}{u_2},\frac{0.5}{u_3},\frac{0}{u_4},\frac{0}{u_5},\frac{0}{u_6},\frac{0}{u_7}\right\} \\ &\vdots \end{aligned} \tag{14}$$

$$A_6 = \left\{\frac{0}{u_0}, \frac{0}{u_1}, \frac{0}{u_2}, \frac{0}{u_3}, \frac{0}{u_4}, \frac{0.5}{u_5}, \frac{1}{u_6}, \frac{0.5}{u_7}\right\}$$

$$A_7 = \left\{\frac{0}{u_0}, \frac{0}{u_1}, \frac{0}{u_2}, \frac{0}{u_3}, \frac{0}{u_4}, \frac{0}{u_5}, \frac{0.5}{u_6}, \frac{1}{u_7}\right\}$$

4. Fuzzify

**Algorithm 4: Fuzify**

```
def fuzzify(df,mtx_interval,data_length):
    mtx_fuzzify = []
    for i in range (data_length):
        curr_data = (df[flag_file][i])
        for j in range (len(mtx_interval)):
            if (j == 0):
                if (mtx_interval[j][0] <= curr_data) and (mtx_interval[j][1] >= curr_data):
                    mtx_fuzzify.append([i,j])
            else:
                if (mtx_interval[j][0] < curr_data) and (mtx_interval[j][1] >= curr_data):
                    mtx_fuzzify.append([i,j])
    return(mtx_fuzzify)
```

5. FLR

In determining the FLR (Algorithm 5), there is a state reduction because, at $t = 0$, there is no transition from the previous state.

**Algorithm 5: FLR**

```
def flr(mtx_fuzzify):
    mtx_flr = []
    for i in range (len(mtx_fuzzify)-1):
        mtx_flr.append((i+1, mtx_fuzzify[i][1],mtx_fuzzify[i+1][1]))
    return (mtx_flr)
```

From Algorithm 4 and Algorithm 5, the results of data fuzzification and FLR can be seen in Table 4. For example, data at $t = 2$ is included in state $A_3$ because it is in the range of interval (202;250], see Table 3, then a transition occurs (FLR) from state $A_2$ to state $A_3$ ($A_2 \rightarrow A_3$).

**Table 4**
Fuzzify and FLR

| $t$ | Month | $Y(t)$ | Fuzzify | FLR |
|---|---|---|---|---|
| 0 | Jan-16 | 181 | $A_2$ | - |
| 1 | Feb-16 | 163 | $A_2$ | $A_2 \rightarrow A_2$ |
| 2 | Mar-16 | 227 | $A_3$ | $A_2 \rightarrow A_3$ |
| 3 | Apr-16 | 209 | $A_3$ | $A_3 \rightarrow A_3$ |
| 4 | May-16 | 236 | $A_3$ | $A_3 \rightarrow A_3$ |
| 5 | Jun-16 | 218 | $A_3$ | $A_3 \rightarrow A_3$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 74 | Mar-22 | 353 | $A_6$ | $A_5 \rightarrow A_6$ |
| 75 | Apr-22 | 388 | $A_6$ | $A_6 \rightarrow A_6$ |
| 76 | May-22 | 396 | $A_7$ | $A_6 \rightarrow A_7$ |
| 77 | Jun-22 | 439 | $A_7$ | $A_7 \rightarrow A_7$ |
| 78 | Jul-22 | 388 | $A_6$ | $A_7 \rightarrow A_6$ |

6. FLRG

**Algorithm 6: FLRG**

```
def flrg(mtx_flr,K_rounded_up):
    int_K_rounded_up = int(K_rounded_up)
    mtx_flrg = [[] for j in range(int_K_rounded_up) ]
    for i in range (len(mtx_flr)):
        temp = mtx_flr[i][1]
        mtx_flrg[temp].append(mtx_flr[i][2])
    return (mtx_flrg)
```

From Algorithm 6, the FLRG results are in Table 5. For example, state $A_0$ transitions to state $A_0$ eight times and to state $A_1$ six times; in another example, state $A_4$ only transitions to states $A_3$ and $A_5$.

**Table 5**
FLRG

| State $Y(t-1) \rightarrow$ State $Y(t)$ |
|---|
| $A_0 \rightarrow 8(A_0), 6\,(A_1)$ |
| $A_1 \rightarrow 6(A_0), 3(A_1), 5(A_2), 2(A_3)$ |
| $A_2 \rightarrow 5(A_1), 9(A_2), 5(A_3)$ |
| $A_3 \rightarrow 2(A_1), 4(A_2), 13\,(A_3), 2(A_4)$ |
| $A_4 \rightarrow A_3, A_5$ |
| $A_5 \rightarrow A_5, A_6$ |
| $A_6 \rightarrow A_6, A_7$ |
| $A_7 \rightarrow A_6\,, A_7$ |
| $A_0 \rightarrow 8(A_0), 6\,(A_1)$ |
| $A_1 \rightarrow 6(A_0), 3(A_1), 5(A_2), 2(A_3)$ |
| $A_2 \rightarrow 5(A_1), 9(A_2), 5(A_3)$ |
| $A_3 \rightarrow 2(A_1), 4(A_2), 13\,(A_3), 2(A_4)$ |

7. The Markov chain transition probability matrix

In Algorithm 7, was built "arrs" function which is the number of transitions from FRLG.

**Algorithm 7: The Markov chain transition probability matrix**

```
1:  generate_big_mtx(mtx_flrg):
2:  arrs = np.array(mtx_flrg)
3:  idx = [np.ones(len(a))*i for i, a in enumerate(arrs)]
4:  (rows, cols), table = npi.count_table(np.concatenate(idx), np.concatenate(arrs))
5:  table = table / table.sum(axis=1, keepdims=True)
6:  return (table)
```

The Markov chain transition probability matrix is obtained in Eq. (15). For example, to determine the transition probability $p_{00}$, $p_{00} = \frac{M_{00}}{M_0} = \frac{8}{8+6} = \frac{8}{14} = 0.57$, where $M_{00}$ is the number of transitions from state $A_0$ to $A_0$ ($A_0 \rightarrow A_0$), and $M_0$ is the complete transition that occurs in $A_0$, which can be seen in Table 5.

$$P = \begin{bmatrix} 0.57 & 0.43 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.37 & 0.19 & 0.31 & 0.13 & 0 & 0 & 0 & 0 \\ 0 & 0.26 & 0.48 & 0.26 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0.19 & 0.61 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{15}$$

8. Forecast

There is a reduction in the state when determining forecast and adjustment value because, at $t = 0$, there is no transition from the previous state, so there is no FLR.

**Algorithm 8: Forecast**

```
1:  def initial_forecast(df,data_length,mtx_flr,mtx_flrg,big_mtx,mid_point):
2:      mtx_initial_forecast = []
3:      new_mtx_flr = [(0,0,0)]
4:      for i in mtx_flr:
5:          new_mtx_flr.append(i)
6:      for i in range (int(data_length)+1):
7:          if (i != 0):
8:              ii = (new_mtx_flr[i][1])
9:              if len(set(mtx_flrg[ii])) == 1:
10:                 to_append = mid_point[new_mtx_flr[i][1]]
11:                 mtx_initial_forecast.append((i,to_append))
12:             elif len(set(mtx_flrg[ii])) > 1: #satu ke banyak
13:                 idx = new_mtx_flr[i][1]
14:                 temp_sum = df[flag_file][i-1] * big_mtx[idx][idx]
15:                 for j in range (len(big_mtx[idx])):
16:                     if (j != idx):
17:                         temp_sum += mid_point[j] * big_mtx[idx][j]
18:                 to_append = temp_sum
19:                 mtx_initial_forecast.append((i,to_append))
20:             else:
21:                 to_append = mid_point[new_mtx_flr[i][1]]
22:                 mtx_initial_forecast.append((i,to_append))
23:     return (mtx_initial_forecast)
```

## 9. Adjustment Value

**Algorithm 9: Adjustment value**

```
def generate_mtx_dt(mtx_flr,interval_class):
    mtx_dt = []
    for i in range (len(mtx_flr)):
        elem = mtx_flr[i]
        selisih = elem[2] - elem[1]
        if (selisih) == 0:
            mtx_dt.append(0)
        else:
            mtx_dt.append(interval_class/2 * selisih)
    return (mtx_dt)
```

## 10. Final forecast

**Algorithm 10: Final forecast**

```
def generate_mtx_final_forecast(df,mtx_initial_forecast,mtx_dt):
    mtx = [df[flag_file][0]]
    for i in range (len(mtx_initial_forecast)):
        temp = abs(mtx_initial_forecast[i][1] + mtx_dt[i])
        mtx.append(temp)
    return (mtx)
```

## 11. Next period

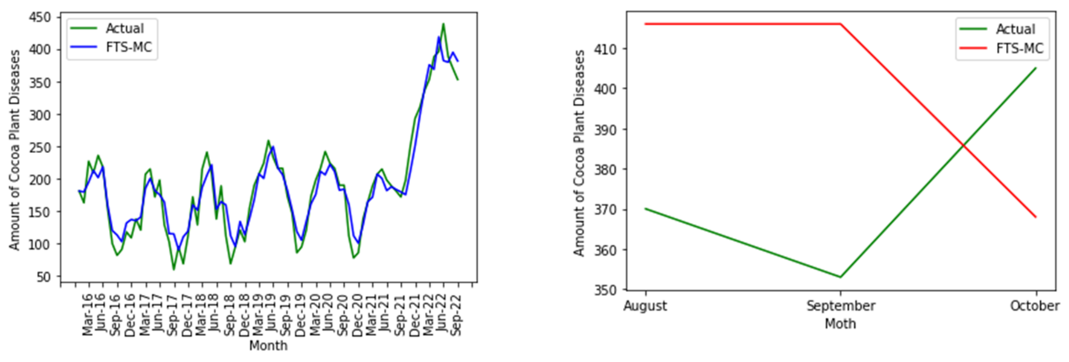
**Algorithm 11: Next Period**

```
next_time = mid_point[mtx_flr[data_length-2][2]
print("Next time value: "+str(next_time))
```

The middle value of the previous state is used to determine the next period. From Algorithm 11, the final forecast for the next period (August 2022) is 416. To forecast the cocoa plant diseases in September and October 2022, re-enter the data by entering the actual value of the month and then run the program. Table 6 shows the forecast result, adjustment value, and final forecast. If depicted in graphical form, it can be seen in Fig. 2.

**Table 6**
Forecasting results of the FTS-MC model for cocoa plant diseases in Bendungan District

| $t$ | Month | $Y(t)$ | $F(t)$ | $D_t$ | $F'(t)$ |
|---|---|---|---|---|---|
| 0 | Jan-16 | 181 | - | - | 181 |
| 1 | Feb-16 | 163 | 179.68 | 0 | 179.68 |
| 2 | Mar-16 | 227 | 171.16 | 23.75 | 194.9 |
| 3 | Apr-16 | 209 | 212.9 | 0 | 212.9 |
| 4 | May-16 | 236 | 201.76 | 0 | 201.76 |
| 5 | Jun-16 | 218 | 218.48 | 0 | 218.48 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 73 | Feb-22 | 336 | 339 | 0 | 339 |
| 74 | Mar-22 | 353 | 352 | 23.75 | 375.75 |
| 75 | Apr-22 | 388 | 384.5 | 0 | 384.5 |
| 76 | May-22 | 396 | 402 | 23.75 | 425.75 |
| 77 | Jun-22 | 439 | 382 | 0 | 382 |
| 78 | Jul-22 | 388 | 403.5 | -23.75 | 379.75 |
| 79 | Aug-22 | 370 | - | - | 416 |
| 80 | Sep-22 | 353 | - | - | 416 |
| 81 | Oct-22 | 405 | - | - | 368 |



(a) (b)

**Fig. 2.** Graph of comparison of actual data with forecasting results of the FTS-MC Model for forecasting the number of diseases cocoa plant in Bendungan District. (a) data from January 2016-October 2022; (b) data from August-October 2022.

12. MAPE value

**Algorithm 12: MAPE**

```
def generate_mtx_mape(df,mtx_final_forecast):
    mtx = [0]
    actual = float(input("Enter The Actual Value: "))
    forecast = mid_point[mtx_flr[data_length-2][2]]
    for i in range (1,len(mtx_final_forecast)):
        data = df[flag_file][i]
        temp = abs((data-mtx_final_forecast[i])/data)
        mtx.append(temp)
```

The MAPE result (

Table **7**) averaged 12.63% which is forecasting cocoa plant diseases in Bendungan District is a good forecast (based on Table 1).

**Table 7**
MAPE value

| Data | Forecast Month | $Y(t)$ | $F'(t)$ | MAPE |
|---|---|---|---|---|
| January 2016-July 2022 | August 2022 | 370 | 416 | 12.73% |
| January 2016-August 2022 | September 2022 | 353 | 416 | 12.58% |
| January 2016-September 2022 | October 22 | 405 | 368 | 12.58% |
| | **MAPE Average** | | | 12.63% |
| | **Forecast Accuracy** | | | Good Forecast |

## 4. Conclusion

Supporting libraries are needed in building a python script for the FTS-MC model, namely pandas, NumPy, math, NumPy indexed, warnings, and sys. The process of looping and if else statements are also used to define the movement of each state. The Python script for the FTS-MC model built in this study can simplify and speed up the calculation process. Determining the forecast for the next period only takes less than a minute. In addition, constructing a python script for the FTS-MC model can reduce calculation or human errors.

The drawback of the python script for the FTS-MC model used in this study is that it requires data to be input in .csv format. In addition, forecasting several future periods requires data input again. Hopefully, this script can be developed so that the forecasting process for several successive periods does not require repeated data input. Moreover, it can be set as a web application to make it more user-friendly.

Based on Fig. **2**(a), actual and forecast data have almost the same data patterns, while in Fig. **2**(b), actual and forecast data have data patterns that tend to be different. Therefore, forecasting using the FTS-MC model is valid for the short term. However, the accuracy of the FTS-MC model using MAPE in predicting the number of cocoa plant diseases in Bendungan District is included in the good forecast (based on Table 1) because it has an average MAPE value of 12.63%.

## Acknowledgments

The authors thank the Rector, Directorate of Research and Community Service (DRPM), and Studies Center of Modeling and Computation Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran. This research was funded by the Padjadjaran Excellence Fastrack Scholarship (BUPP), Academic Leadership Grant (ALG) grant number 2203/UN6.3.1/PT.00/2022, and supported by the international Consortium RISE_SMA project 2019-2024.

## References

Alyousifi, Y., Othman, M., Husin, A., & Rathnayake, U. (2021). A new hybrid fuzzy time series model with an application to predict PM10 concentration. *Ecotoxicology and Environmental Safety*, *227*. https://doi.org/10.1016/j.ecoenv.2021.112875

BBPPTP. (2023). https://simopt-bbpptp-surabaya.net. (Online: Accessed January 1, 2023)

Canlas, F. Q., Nair, S., & Paat, I. D. (2022). Exploring COVID-19 Vaccine Side Effects: A Correlational Study Using Python. *Procedia Computer Science*, *201*(C), 752–757. https://doi.org/10.1016/j.procs.2022.03.102

Cascio, W. F., & Montealegre, R. (2016). How Technology Is Changing Work and Organizations. *Annual Review of Organizational Psychology and Organizational Behavior*, 3, pp. 349–375. https://doi.org/10.1146/annurev-orgpsych-041015-062352

Chen, S.M. (1996). Forecasting enrollments based on fuzzy time series. *Fuzzy Sets and Systems*, 81.

Hashim, H. (2018). Application of Technology in the Digital Era Education. *International Journal of Research in Counseling and Education*, *1*(2), 1. https://doi.org/10.24036/002za0002

Kanewala, U., & Bieman, J. M. (2014). Testing scientific software: A systematic literature review. *Information and Software Technology*, 56, 10, 1219–1232. https://doi.org/10.1016/j.infsof.2014.05.006

Kemenperin. (2023). https://www.kemenperin.go.id. (Online: Accessed January 1, 2023)

Langtangen, H. P. (2008). *Python Scripting for Computational Science*, 3rd ed. Berlin Heidelberg: Springer-Verlag.

Lawrence, K. D., Klimberg, R. K., & Lawrence, S. M. (2009). *Fundamentals of Forecasting Using Excel*. Industrial Press Inc.

Leon-Alcaide, P., Rodriguez-Benitez, L., Castillo-Herrera, E., Moreno-Garcia, J., & Jimenez-Linares, L. (2020). An evolutionary approach for efficient prototyping of large time series datasets. *Information Sciences*, 511, 74–93. https://doi.org/10.1016/j.ins.2019.09.044

Listewnik, K. J., & Aftewicz, K. (2023). Rotary 3D Magnetic Field Scanner for the Research and Minimization of the Magnetic Field of UUV. *Sensors*, *23*(1). https://doi.org/10.3390/s23010345

Lutz, M. (2013). *Learning Python*. Sebastopol: O'REILLY.

Paper, D. (2021). *TensorFlow 2.x in the Colaboratory Cloud : An Introduction to Deep Learning on Google's Cloud Service*. Apress. https://doi.org/https://doi.org/10.1007/978-1-4842-6649-6 ISBN-13

Petrelli, M. (2021). *Introduction to Python in Earth Science Data Analysis : From Descriptive Statistics to Machine Learning*. Springer: Springer Textbooks in Earth Sciences, Geography and Environment. https://doi.org/https://doi.org/10.1007/978-3-030-78055-5

Sayama, H., Pestov, I., Schmidt, J., Bush, B. J., Wong, C., Yamanoi, J., & Gross, T. (2013). Modeling complex systems with adaptive networks. *Computers and Mathematics with Applications*, *65*(10), 1645–1664. https://doi.org/10.1016/j.camwa.2012.12.005

Song, Q., & Chissom, B. S. (1993). Forecasting enrollments with fuzzy time series - Part I. *Fuzzy Sets and Systems*, *54*(1), 1–9. https://doi.org/10.1016/0165-0114(93)90355-L

Song, Q., & Chissorn, B. S. (1994). Forecasting enrollments with fuzzy time series-part II. In *Fuzzy Sets and Systems*, 62.

Tsaur, R. C. (2012). A fuzzy time series-Markov chain model with an application to forecast the exchange rate between the Taiwan and us Dollar. *International Journal of Innovative Computing, Information and Control*, *8*(7 B), 4931–4942.

Wang, Y., Chen, Z., Tian, S., Zhou, S., Wang, X., Xue, L., & Wu, J. (2022). Convolutional Neural Network-Based ECG-Assisted Diagnosis for Coal Workers. *Environmental Research and Public Health*, 20, 17. https://doi.org/10.3390/ijerph

Zadeh, L. A. (1996). Fuzzy Sets. *Information and Control,* 394–432. https://doi.org/10.1142/9789814261302_0021

Zhu, T., & Liu, G. (2023). A Novel Hybrid Methodology to Study the Risk Management of Prefabricated Building Supply Chains: An Outlook for Sustainability. *Sustainability, 15*(1). https://doi.org/10.3390/su15010361

Zwicke, F., Knechtges, P., Behr, M., & Elgeti, S. (2016). Automatic implementation of material laws: Jacobian calculation in a finite element code with TAPENADE. *Computers and Mathematics with Applications*, *72*(11), 2808–2822. https://doi.org/10.1016/j.camwa.2016.10.010