Theta: Journal of Statistics vol. 1, no. 1, pp. 16-26, 2025

Theta: Journal of Statistics

Journal homepage: www.jurnal.untirta.ac.id/index.php/tjs

Forecasting the Open Unemployment Rate in Banten Province Using the FB Prophet Method in Python Programming Language

Ferdian Bangkit Wijaya[a] *, Deananta Pramudia Putra[b], Mahsa Azzahra[a] , Faula Arina[a], Fajri Ikhsan[c]

[a] *Department of Statistics, Universitas Sultan Ageng Tirtayasa, Jl. Jenderal Sudirman KM 3 Cilegon Banten, Indonesia*
[b] *Kementerian Komunikasi dan Digital Republik Indonesia*
[c] *Department of Chemistry, Universitas Negeri Padang, Jl. Prof. Dr. Hamka Air Tawang Padang, Padang, Indonesia*
*Corresponding Author: ferdian.bangkit@untirta.ac.id

INFORMATION

Article information:
Submitted: 14 February 2025
Revised: 28 March 2025
Accepted: 31 March 2025
Available Online: 31 March 2025

Keywords:
Open Unemployment Raye; Prophet; Python; Forecasting Model; TPT Banten

ABSTRACT

The Open Unemployment Rate is a key indicator in measuring labor market imbalances, reflecting the economic dynamics of a region. Banten Province has consistently ranked among the top three provinces with the highest Open Unemployment Rate in Indonesia over the past decade, indicating structural challenges in employment. To address this issue, a forecasting model is needed to provide accurate predictions that support more effective labor policy planning. This study uses the Prophet method, an additive regression approach developed by Facebook, to forecast the Open Unemployment Rate in Banten Province over the next 10 semesters (February 2025-August 2029). The data used is sourced from the Statistics Indonesia (BPS) for the period 2005-2024, collected every semester (February and August). The model's performance is evaluated using the Mean Absolute Percentage Error (MAPE) as the primary evaluation metric. The results show that the Prophet model effectively captures trend and seasonal patterns. With a MAPE value of 5.3910%, the model demonstrates very good accuracy (MAPE < 10%), making it suitable for medium-term forecasting. The predictions indicate a downward trend in the Open Unemployment Rate in Banten over the next five years. The conclusion of this study suggests that the Prophet model can be a reliable tool for projecting the Open Unemployment Rate and supporting labor policy planning in Banten. Future research is expected to incorporate external factors or use hybrid modeling approaches to improve prediction accuracy.

INTRODUCTION

The Open Unemployment Rate (TPT) is the percentage of the population who are unemployed and actively seeking work in relation to the total labor force over a certain period [1]. This indicator is used to measure imbalances in the labor market, reflecting the economic dynamics of a region. As a result, it is often used as a reference in the formulation of labor policies and economic development strategies.

ISSN xxxxxx print/ xxxx-xxxx online © 20XX
DOI: http://dx.doi.org/10.62870/tjs.v1i1.31309

According to data from the National Labor Force Survey (SAKERNAS) conducted by the Statistics Indonesia (BPS), the Province of Banten has consistently ranked among the top three provinces with the highest Open Unemployment Rate in Indonesia over the past decade. The monthly SAKERNAS publications in August show that the TPT in Banten has never been below the national average. A higher Open Unemployment Rate indicates an increasing number of unemployed individuals, which can negatively affect the well-being of the population [2]. This demonstrates that the labor sector in Banten faces significant structural challenges, which may impact both economic and social stability in the region.

In the context of ASEAN and Asian countries, the TPT standards vary based on the level of economic development. Developed countries such as Japan and South Korea have relatively low TPT, ranging from 2-4%, while developing countries like Indonesia, Malaysia, and Thailand have higher TPT, ranging from 4-6%. This comparison suggests that to achieve economic stability, the Province of Banten must reduce its TPT to a more competitive level [3]. To address the labor challenges in the Province of Banten, a forecasting model that can predict the TPT for the coming years is needed. With accurate forecasting, local governments can formulate more effective policies to address unemployment, including workforce training programs, investment incentives, and the development of the creative economy sector.

Previous studies have applied the Triple Exponential Smoothing method to model the TPT in Banten, resulting in a Mean Absolute Percentage Error (MAPE) of 8.85% [4]. Although this method produced a MAPE value of less than 10%, it is still less effective in capturing long-term trends and accommodating outliers often caused by economic fluctuations. Therefore, a more adaptive forecasting model is required to improve the accuracy of TPT predictions in Banten. This study will use the FB Prophet method to address the limitations of the previously used method.

RESEARCH METHODS

This study uses data on the Open Unemployment Rate (TPT) obtained from the official website of the Statistics Indonesia (BPS), with the most recent data update in August 2024. The data specifically focuses on the Province of Banten, covering the period from 2005 to 2024. This data is collected periodically every semester (February and August).

The forecasting model used in this study is Prophet, an additive regression method developed by Facebook. Prophet is designed to handle long-term trends as well as recurring seasonal patterns with a flexible approach. Additionally, the Prophet model can handle outliers by considering change points, seasonality with Fourier order, and trend changes by adding regressors into the model, enabling it to adapt to values that change significantly [5]. The data is divided into two main groups: the Training Set, which covers the period from 2005 to 2021 and is used for model training and creation, and the Testing Set, which covers the period from 2022 to 2024 and is used to evaluate the model's accuracy. The model created is then used to forecast or project the TPT for the next 10 semesters (February 2025 – August 2029). The model's performance will be evaluated using six metrics: Mean Error (ME), Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Percentage Error (MPE), and Mean Absolute Percentage Error (MAPE). The MAPE will be the primary evaluation metric in this study. If the MAPE result is < 10%, the model is considered excellent and can be used for future forecasting [6]. In general, the Prophet model can be written as follows [7]:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t) \tag{1}$$

where,
$y(t)$ is the forecasted value at time t,
$g(t)$ is the trend component,
$s(t)$ is the seasonal component,
$h(t)$ is the holiday effect,
$\varepsilon(t)$ is the noise or error.

In Prophet modeling, there are four options for tuning the trend/growth parameter:

a. Non-linear Saturating Growth

The growth component aims to understand how a population evolves. Facebook designed the growth component to closely resemble an ecological system. Growth is typically modeled using the logistic growth model in its simplest form, as follows:

$$g(t) = \frac{C}{1+\exp\left(-k(t-m)\right)} \tag{2}$$

Where,

$g(t)$ is the growth or trend component at time-t,
C is the carrying capacity,
k is the growth rate,
m is the offset parameter.

Alternatively, the piecewise logistic growth model can be used:

$$g(t) = \frac{C(t)}{1+\exp\left(-(k+a(t)^T\delta)(t-(m+a(t)^T\delta)\right)} \tag{3}$$

Where,

$g(t)$ is the growth or trend component at time-t,
$C(t)$ is the time-varying capacity,
$k + a(t)^T\delta$ is the rate at time-t,
δ is a vector of rate adjustments,
$a(t)$ is a vector of $\{0,1\}^s$,
T is transpose of vector.

b. Linear Trend with Changepoints

Changepoints are moments when the data direction changes. For forecasting without growth and a piecewise constant rate of growth, the trend model becomes:

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\,\gamma) \tag{4}$$

Where,

$g(t)$ is the growth or trend component at time-t,
k is the growth rate,
$a(t)$ is a vector of $\{0,1\}^s$,
δ is a vector of rate adjustments,
m is the offset parameter,
T is transpose of vector,
γ is adjusted to $-s_j\delta_j$ for continuity.

The changepoint prior scale is used to regulate the trend's flexibility, addressing overfitting and underfitting issues [8].

c. Automatic Changepoint Selection

This model is used when the times of trend changes are known. This process is carried out automatically by filtering candidates and selecting based on Equations 3 and 4.

d. Trend Forecast Uncertainty
When the model is extrapolated to make forecasts, the trend will have constant values. A generative model forward can be used to estimate the uncertainty in the trend forecast. Future changepoints are taken randomly, so the average frequency of changepoints fits the historical data.

For determining seasonality or seasonal patterns, the Fourier series is often used to provide a flexible model of periodic effects. The equation for an arbitrary smooth seasonal effect is:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \cos\left(\frac{2\pi nt}{P}\right) \right) \quad (5)$$

Where,
Parameter $\beta = [a_1, b_1, \ldots, a_n, b_n]^T$

$a_n = \frac{1}{P} \int_{-P}^{P} f(x) \cos\left(\frac{2\pi nt}{P}\right) dx$

$b_n = \frac{1}{P} \int_{-P}^{P} f(x) \sin\left(\frac{2\pi nt}{P}\right) dx$

Parameter selection can be automated using the AIC model selection procedure. The seasonality component can be set as either an additive model or a multiplicative model [9]. In the case of multiplicative seasonality, the growth component *g(t)* and *seasonality s(t)* will undergo log transformation [10].

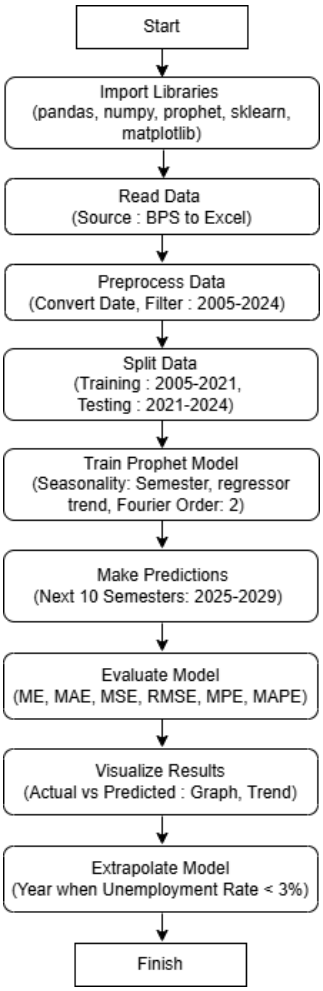

Figure 1. Prophet Modeling Flowchart with Python

The process of modeling and forecasting using Prophet is implemented using the Python programming language and executed within the Visual Studio Code environment. The programming workflow used in this study is illustrated in Figure 1.

RESULTS AND DISCUSSION

The process of modeling and forecasting the Open Unemployment Rate (TPT) in the Province of Banten using the FB Prophet method was carried out through several systematic stages. The process began with importing the required libraries into the Python environment, namely pandas for data management, manipulation, and analysis, NumPy for numerical computation, Prophet as the time-series forecasting model, sklearn.metrics for model evaluation (specifically used for calculating MAE and MSE), and matplotlib for visualizing graphs. The code lines for importing libraries in the Python Visual Studio Code environment can be seen in Figure 2.

```
import pandas as pd
import numpy as np
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
import matplotlib.pyplot as plt
```

Figure 2. Code Lines for Importing Libraries with Python

If the libraries are not installed, they must first be installed through the terminal or command prompt using the code lines shown in Figure 3.

```
pip install pandas numpy prophet scikit-learn matplotlib
```

Figure 3. Code Lines for Installing Libraries with Python

After importing the libraries, the unemployment rate data from the Central Statistics Agency's website for the 2025 access year is saved in Excel format and then read. The code lines for reading the Excel file can be seen in Figure 4.

```
# Path to the Excel file
file_path = r"C:\.....\DATA_TPT_BANTEN.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)
```

Figure 4. Code Lines for Reading an Excel File with Python

Once the Excel data is successfully read, preprocessing is performed, which includes converting the date format to meet the requirements of the Prophet library in Python. The data used only spans from 2005 to 2024, so date filtering is necessary. The code lines for this filtering process can be seen in Figure 5.

```
# Menggabungkan Tahun dan Bulan menjadi satu kolom datetime
month_mapping = {
    'Januari': 'January', 'Februari': 'February', 'Maret': 'March', 'April': 'April',
    'Mei': 'May', 'Juni': 'June', 'Juli': 'July', 'Agustus': 'August',
    'September': 'September', 'Oktober': 'October', 'November': 'November', 'Desember': 'December'
}
df['Bulan'] = df['Bulan'].map(month_mapping)
df['date'] = pd.to_datetime(df['Tahun'].astype(str) + ' ' + df['Bulan'], format='%Y %B')

# Filter data mulai dari tahun 2005
df = df[df['date'] >= '2005-01-01']

# Mengatur format data untuk Prophet
df_prophet = df[['date', 'TPT_Banten']].rename(columns={'date': 'ds', 'TPT_Banten': 'y'})
```

Figure 5. Code Lines for Preprocessing, Date Conversion, and Date Filtering with Python

After preprocessing, the data is divided into two subsets: the training data (2005–2021), used to build the model, and the testing data (2022–2024), used to evaluate the prediction accuracy of the model. The code lines for creating these data subsets can be seen in Figure 6.

```
# Membagi data menjadi train dan test set (Menggunakan data hingga 2024 untuk training)
train = df_prophet[df_prophet['ds'] < '2022-01-01']
test = df_prophet[(df_prophet['ds'] >= '2022-01-01') & (df_prophet['ds'] <= '2024-12-31')]
```

Figure 6. Code Lines for Data Subsetting with Python

Following data separation, exploratory data analysis is conducted by examining patterns and trends for each year or semester. The code lines for creating this exploratory data visualization can be seen in Figure 7.

```
# Plot deskriptif tren dari tahun 2005 hingga 2024
plt.figure(figsize=(12, 6))
plt.plot(df_prophet['ds'], df_prophet['y'], label="TPT Banten", marker='o', linestyle='-')
plt.xlabel("Year")
plt.ylabel("TPT Banten")
plt.title("Descriptive Trend of TPT Banten (2005-2024)")
plt.legend()
plt.show()
```

Figure 7. Code Lines for Plotting Data Series with Python

In Figure 8, two trend patterns in the training data are observed: one from 2005 to 2011 and another from 2012 to 2021. Consequently, an adaptive time regressor component was added to the Prophet model.

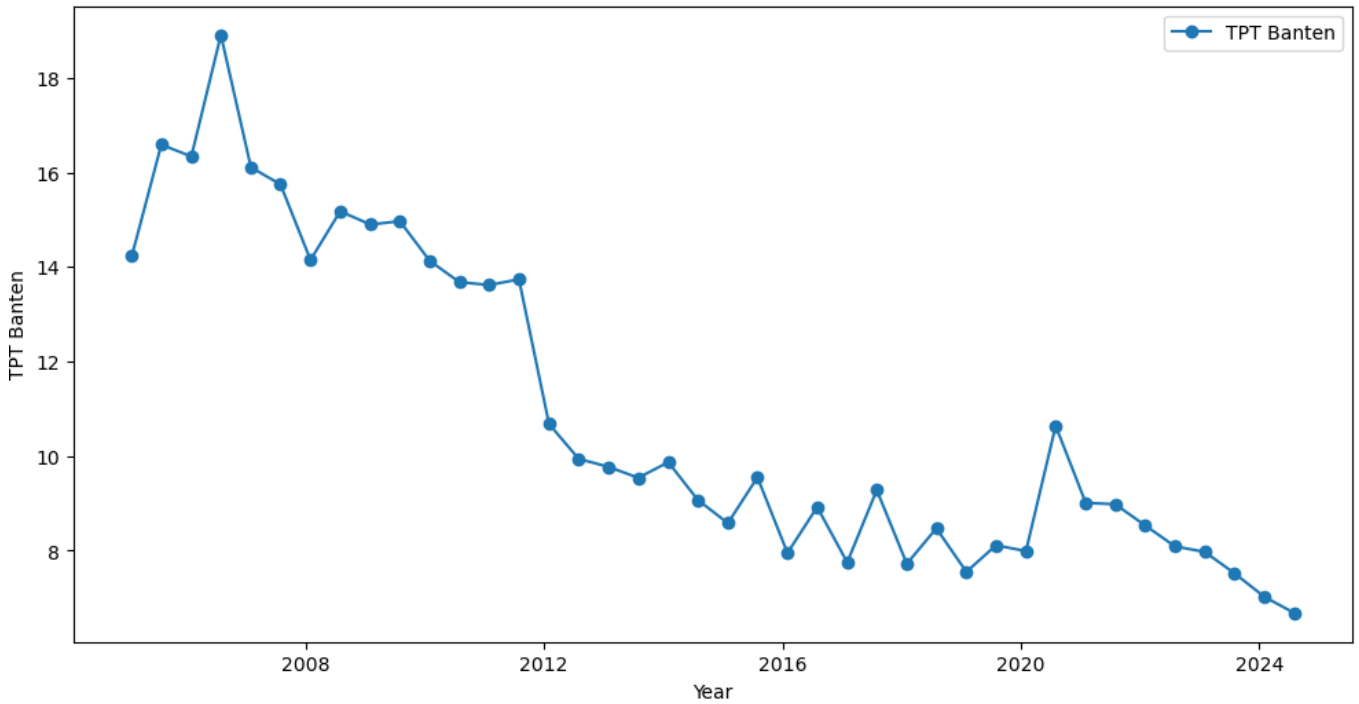

Figure 8. Unemployment Rate Data Series of Banten 2005–2024

In addition to adding the adaptive time regressor component, seasonal patterns for each semester were also observed. As shown in Figure 8, the data exhibits rising and falling patterns, leading to the Prophet model being trained using seasonal parameters for each semester (February and August) with a Fourier Order of 2 to capture seasonal patterns in the data. The seasonal component chosen was based on

the semester, not the year, month, week, or day, as the pattern fluctuates seasonally. The code lines for creating and training the Prophet model can be seen in Figure 9.

```
# Membuat dan melatih model Prophet dengan seasonality per semester
model = Prophet(yearly_seasonality=False, weekly_seasonality=False, daily_seasonality=False)
model.add_seasonality(name='semester', period=0.5, fourier_order=2)
model.add_regressor('trend_2005_2011')
model.add_regressor('trend_2012_2024')
model.fit(train)
```

Figure 9. Code Lines for Creating and Training the Prophet Model with Python

Once the model was created, forecasting was performed for the next 10 periods (February 2025 to August 2029) to obtain projections of the unemployment rate. The code lines for this forecasting or projection process can be seen in Figure 10.

```
# Melakukan prediksi untuk 5 tahun ke depan (10 semester)
future_periods = 10  # 10 semester (5 tahun)
future = model.make_future_dataframe(periods=future_periods * 2, freq='6M')  # 10 semester (5 tahun)
future['trend_2005_2011'] = ((future['ds'] >= '2005-01-01') & (future['ds'] <= '2011-12-31')).astype(int)
future['trend_2012_2024'] = ((future['ds'] >= '2012-01-01') & (future['ds'] <= '2024-12-31')).astype(int)

# Ensure future includes the test dates
future = pd.concat([future, test[['ds', 'trend_2005_2011', 'trend_2012_2024']]]).drop_duplicates().sort_values('ds').reset_index(drop=True)

forecast = model.predict(future)
```

Figure 10. Code Lines for Forecasting or Projecting the Next 10 Semesters with Python

The results of the 10-semester forecast are as follows:

Table 1. Forecasting Results/Projections for the Next 10 Semesters

Period	Estimated TPT (%)	Minimum Estimate (%)	Maximum Estimate (%)
February 2025	6.764309	5.460869	8.055364
August 2025	6.652490	5.360154	7.894465
February 2026	6.542494	5.256962	7.831959
August 2026	6.430675	5.181750	7.728252
February 2027	6.320679	5.007762	7.649004
August 2027	6.208860	4.910097	7.515059
February 2028	6.098256	4.738835	7.362857
August 2028	5.986437	4.757324	7.269946
February 2029	5.876441	4.586470	7.178969
August 2029	5.764622	4.429874	7.068574

Based on Table 1, the prediction range, i.e., Minimum Estimate (%) and Maximum Estimate (%), indicates uncertainty in the model's projections, where the prediction results fall within a specific interval, calculated based on probability distribution. The point estimate can be seen in the "Estimated TPT (%)" column.

After performing the forecasting, it is important to reassess whether the forecast is valid and usable. Model evaluation is conducted by calculating several evaluation metrics. The metrics used are Mean Error (ME), Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Percentage Error (MPE), and Mean Absolute Percentage Error (MAPE). These metrics are used to assess the accuracy of the model in predicting the TPT based on the testing data. The code lines for model evaluation can be seen in Figure 11.

```python
# Mengambil hasil prediksi untuk data uji
test_pred = forecast[forecast['ds'].isin(test['ds'])]['yhat'].values  # Mengambil prediksi pada rentang test
actual = test['y'].values  # Data aktual dari test set

# Evaluasi model
me = np.mean(test_pred - actual)  # Mean Error (ME)
mae = mean_absolute_error(actual, test_pred)  # Mean Absolute Error (MAE)
mse = mean_squared_error(actual, test_pred)  # Mean Squared Error (MSE)
rmse = np.sqrt(mse)  # Root Mean Squared Error (RMSE)
mpe = np.mean((test_pred - actual) / actual) * 100  # Mean Percentage Error (MPE)
mape = np.mean(np.abs((test_pred - actual) / actual)) * 100  # Mean Absolute Percentage Error (MAPE)

# Menampilkan hasil evaluasi
evaluation_metrics = {
    "Mean Error (ME)": me,
    "Mean Absolute Error (MAE)": mae,
    "Mean Squared Error (MSE)": mse,
    "Root Mean Squared Error (RMSE)": rmse,
    "Mean Percentage Error (MPE)": mpe,
    "Mean Absolute Percentage Error (MAPE)": mape
}

print("Model Evaluation Metrics:")
for metric, value in evaluation_metrics.items():
    print(f"{metric}: {value:.4f}")
```

Figure 11. Code Lines for Evaluation Metrics with Python

Model evaluation is conducted by comparing the predicted results with the actual data from the testing period (2022–2024). The model's error values are displayed in Table 2.

Table 2. Prophet Model Performance Evaluation Metrics

Evaluation Method	Value
Mean Error (ME)	-0.1108
Mean Absolute Error (MAE)	0.4123
Mean Squared Error (MSE)	0.2135
Root Mean Squared Error (RMSE)	0.4620
Mean Percentage Error (MPE)	-0.9598%
Mean Absolute Percentage Error (MAPE)	5.3910%

From the evaluation results, it can be seen that the RMSE value of 0.4123 indicates a relatively small error in predicting the TPT. Additionally, the MAPE value of 5.3910% shows that the model has a very good prediction error rate, as it is < 10%.

Overall, the Prophet model successfully captures the trend patterns of the TPT. However, there is a slight negative bias in the predictions, as indicated by the ME value of -0.1108. This suggests that the model tends to slightly underestimate the actual values (underfitting).

After model evaluation, the forecasting results are visualized in the form of a graph comparing the actual and predicted data. The code lines for this visualization can be seen in Figure 12.

```python
# Visualisasi hasil prediksi
plt.figure(figsize=(12, 6))
plt.plot(df_prophet['ds'], df_prophet['y'], label="Actual Data", marker='o', linestyle='-')
plt.plot(forecast['ds'], forecast['yhat'], label="Predicted Data", linestyle='dashed', color='red')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], alpha=0.2, color='gray')
plt.axvline(test['ds'].iloc[0], color='black', linestyle='dotted', label="Test Data Start")
plt.xlabel("Year")
plt.ylabel("TPT Banten")
plt.title("Prophet Forecasting for TPT Banten (2022-2029)")
plt.legend()
plt.show()

# Menampilkan hasil prediksi 5 tahun ke depan (2025-2029)
forecast_5_years = forecast[(forecast['ds'] >= '2025-01-01') & (forecast['ds'] <= '2029-12-31')]
print(forecast_5_years[['ds', 'yhat', 'yhat_lower', 'yhat_upper']])

# Visualisasi komponen tren dan musiman
fig = model.plot_components(forecast)
plt.show()
```

Figure 12. Code Lines for Visualizing Forecasting Graphs with Python

The analysis of the Prophet model components shows that the TPT in Banten exhibits a long-term downward trend. Additionally, the seasonal pattern per semester was also identified, indicating fluctuations in the TPT throughout the year. This is shown in Figure 13 and Figure 14.

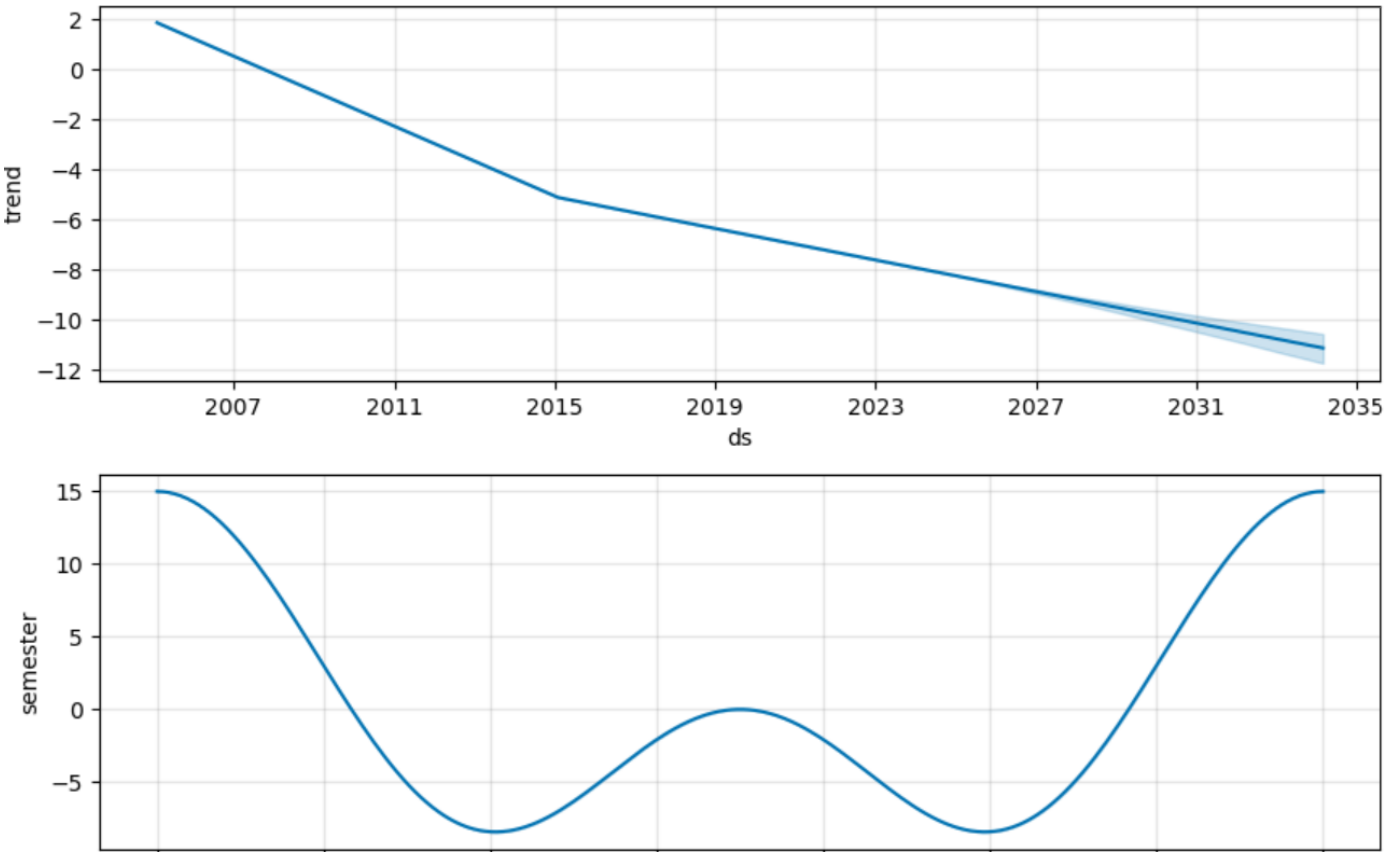

Figure 13. Trend and Seasonal Components of the Prophet Model for Banten's TPT

Specifically, the trend of the TPT in Banten tends to decline over the next five years, which can be associated with various factors, such as labor policies, economic growth, and improvements in workforce skills. The seasonal fluctuations also show a consistent pattern, likely influenced by cyclical factors such as the recruitment season, workforce training, and labor-related policies from both the central and regional governments. The comparison of forecasted and actual values can be seen in Figure 14.

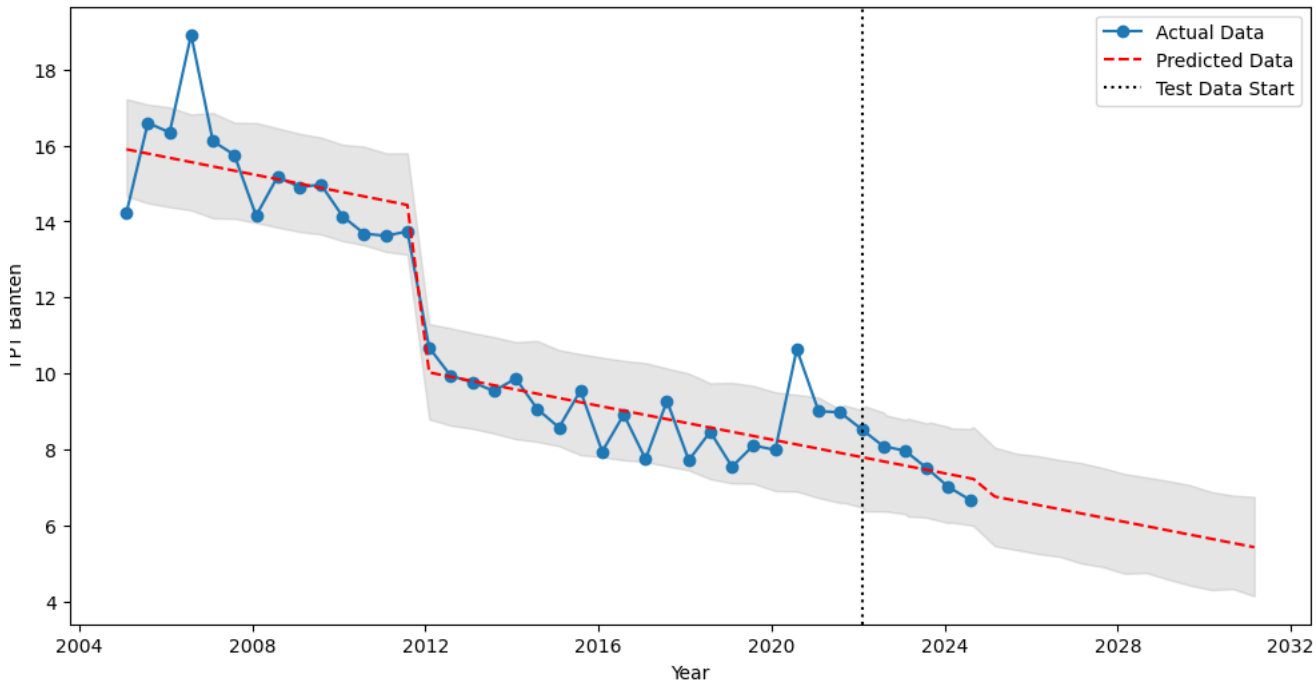

Figure 14. Historical and Forecasting Visualization of Banten's TPT

After obtaining an accurate model capable of forecasting up to 10 semesters ahead, the final step is to conduct a recommendation analysis by identifying the time period when the TPT is predicted to fall below 3% (as per the standards of advanced countries in Asia). This information is useful for policymakers in formulating unemployment reduction strategies. The code lines for insight and recommendations are shown in Figure 15.

```
# Simulasi untuk mencari tahun terakhir di mana yhat berada di bawah 3%
threshold = 3
found = False
current_periods = future_periods

while not found:
    future = model.make_future_dataframe(periods=current_periods * 2, freq='6M')
    future = pd.concat([future, test[['ds']]]).drop_duplicates().sort_values('ds').reset_index(drop=True)
    forecast = model.predict(future)
    below_threshold = forecast[forecast['yhat'] < threshold]
    if not below_threshold.empty:
        last_date_below_threshold = below_threshold['ds'].max()
        print(f"Tahun terakhir di mana yhat berada di bawah {threshold}%: {last_date_below_threshold.year}")
        found = True
    else:
        current_periods += 10  # Tambah 10 semester lagi jika belum ditemukan

if not found:
    print(f"Tidak ada prediksi yhat yang berada di bawah {threshold}%")
```

Figure 15. Code Lines for Insight and Recommendations in Python

Based on the model's recommendation, the TPT is predicted to fall below 3% by the year 2044, assuming ceteris paribus, meaning that all conditions, including economic, policy, and labor circumstances, remain the same as in 2024. It is possible that the 3% threshold could be reached earlier or later, depending on government policies.

CONCLUSION

The Prophet model effectively captures the downward trend of the Open Unemployment Rate (TPT) in the Province of Banten. Fluctuations in the actual data indicate the presence of external factors, which were not explicitly included in this model, such as economic conditions or labor policies. The accuracy of this model is excellent, with a MAPE value of 5.3910%, which is below 10%. This suggests that the

predictions provided can be considered in the formulation of labor policies. The accuracy of the Prophet model can be improved by incorporating external factors. A hybrid approach, combining Prophet with other models, Prophet with additional parameter tuning, or the use of other methods such as Deep Learning LSTM (Long Short-Term Memory) or ARIMAX (Autoregressive Integrated Moving Average – X), can be utilized to enhance the precision of the TPT predictions for the Province of Banten.

REFERENCES

[1] Direktorat Statistik Kependudukan dan Ketenagakerjaan, *Booklet SAKERNAS Agustus 2021*. Badan Pusat Statistik, 2021.

[2] BPS Indonesia, "Tingkat pengangguran terbuka menurut provinsi - tabel statistik," Badan Pusat Statistik Indonesia. [Online]. Available: https://www.bps.go.id/id/statistics-table/2/NTQzIzI=/tingkat-pengangguran-terbuka-menurut-provinsi.html

[3] International Monetary Fund, *World Economic Outlook: Policy Pivot, Rising Threats*, October. Washington, DC, 2024.

[4] R. N. Puspita, "Peramalan Tingkat Pengangguran Terbuka Provinsi Banten Dengan Metode Triple Exponential Smoothing," *Jurnal Lebesgue : Jurnal Ilmiah Pendidikan Matematika, Matematika dan Statistika*, vol. 3, no. 2, pp. 358–366, 2022, doi: 10.46306/lb.v3i2.138.

[5] S. J. Taylor and B. Letham, "Business Time Series Forecasting at Scale," *PeerJ Preprints 5:e3190v2*, vol. 35, no. 8, pp. 48–90, 2017.

[6] F. B. Prakoso, G. Darmawan, and A. Bachrudin, "Penerapan Metode Facebook Prophet Untuk Meramalkan Jumlah Penumpang Trans Metro Bandung Koridor 1," *ARMADA : Jurnal Penelitian Multidisiplin*, vol. 1, no. 3, pp. 133–147, 2023, doi: 10.55681/armada.v1i3.416.

[7] B. H. Winarno, D. Kusumawati, H. A. Triyanto, and B. H. Winarno, "PENERAPAN MACHINE LEARNING (MODEL PROPHET) DALAM PREDIKSI PERMINTAAN PRODUK UNTUK MENGOPTIMALKAN INVENTORI," no. November, pp. 168–174, 2024.

[8] A. Subashini, K. Sandhiya, S. Saranya, and U. Harsha, "Forecasting Website Traffic Using Prophet Time Series Model," *International Research Journal of Multidisciplinary Technovation*, vol. 1, no. 1, pp. 56–63, 2019, doi: 10.34256/irjmt1917.

[9] Y. Ensafi, S. H. Amin, G. Zhang, and B. Shah, "Time-series forecasting of seasonal items sales using machine learning – A comparative analysis," *International Journal of Information Management Data Insights*, vol. 2, no. 1, p. 100058, 2022, doi: 10.1016/j.jjimei.2022.100058.

[10] D. Borges and M. C. V. Nascimento, "COVID-19 ICU demand forecasting: A two-stage Prophet-LSTM approach," *Appl Soft Comput*, vol. 125, p. 109181, 2022, doi: 10.1016/j.asoc.2022.109181.