
The application of Neural Prophet Time Series in predicting rice stock at Rice Stores

Djarot Hindarto^{1)*}, Ferial Hendrata²⁾, Mochamad Hariadi³⁾

¹⁾Prodi Informatika, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional Jakarta, Indonesia

²⁾Prodi Sistem Informasi, Universitas Narotama, Surabaya, Indonesia

³⁾Department of Electrical Engineering, Department of Computer Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

¹⁾djarot.hindarto@civitas.unas.ac.id, ²⁾ferial.hendrata@narotama.ac.id, ³⁾mochar@te.its.ac.id

ABSTRACT

Efficient inventory management and consistent rice supply are pivotal for the sustainability of small-scale food stalls. This research introduces an innovative approach to address this challenge through the Neural Prophet algorithm. By synergizing neural networks with additive regression models, the Neural Prophet captures intricate temporal patterns and trends within rice sales data. Our study evaluates the Neural Prophet's effectiveness in predicting rice sales, specifically for essential food vendors. Leveraging historical sales data from June 2022 to April 2023, the algorithm incorporates seasonality and trends and integrates external events, such as holidays, to heighten prediction precision. Our findings underscore the Neural Prophet's remarkable prowess in forecasting rice sales at primary food kiosks, adeptly discerning data trends and fluctuations, culminating in reliable future sales projections. The model boasts compelling performance metrics: MAE = 12.90, RMSE = 15.80, and Loss = 0.0313. Beyond its technical merits, this research carries significant practical implications, empowering proprietors and suppliers of basic food stalls to streamline inventory management, avert stockouts, and curtail overstocking by harnessing the precision of rice demand forecasting facilitated by the Neural Prophet algorithm.

Keywords: Accurate forecasting; Model Performance; Neural Prophet; Rice; Time Series

INTRODUCTION

Rice is Indonesia's primary agricultural commodity and constitutes the fundamental dietary component for most of the nation's population. The rice production in Indonesia consistently ranks among the highest globally. In 2020, Indonesia was projected to achieve a production output of approximately 54 million tonnes of rice, positioning it as the fifth largest global producer of rice, following China, Vietnam, India, and Bangladesh (Wu et al., 2021). Indonesia exhibits a diverse array of rice varieties that thrive in various geographical locations. Indonesia is known for cultivating several prevalent rice varieties, including Indica, Japonica, and indigenous rice types (ZHOU et al., 2023), including Pandan Wangi, IR 64, and Cianjur rice. Every cultivar has unique characteristics in texture, aroma, and flavor. Rice constitutes a fundamental component of the dietary habits of a significant proportion of the Indonesian population. Despite shifts in dietary patterns among some people, wherein some groups have started adopting alternative meals, rice remains the predominant component in numerous traditional Indonesian culinary practices. Indonesia possesses considerable potential for rice exports, with its capacity to fulfill domestic needs. Indonesia's rice exports are mainly directed towards the Philippines, Malaysia, and Timor Leste, which rank among the country's leading export destinations. The exportation of rice has the potential to contribute significantly to Indonesia's economy and enhance its status as a prominent global rice producer. The Indonesian government is currently making efforts to improve agricultural productivity, particularly in rice cultivation. Implementing several initiatives, including utilizing contemporary agricultural machinery, enhancements in infrastructure, and adopting high-yielding cultivars, can potentially enhance rice productivity (Saputra et al., 2023) and efficiency in Indonesia. It is imperative to remember that various factors, including weather patterns, climate change, and agricultural governmental policies, can influence rice production in Indonesia. Ensuring an adequate and sustainable supply of rice for the people of Indonesia is contingent upon effective management of the agricultural sector, particularly in the realm of rice production.

Rice is a staple food in various countries, including Indonesia. Rice is the primary carbohydrate supply for a significant proportion of the global population, and its demand remains substantial. Grocery stalls are vital

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

establishments where individuals may procure essential food items, including rice, a staple of daily meals. These stalls play a significant role in fulfilling the demand for rice. Managing rice supplies at primary food kiosks frequently presents a challenging task. The lack of precise demand forecasting can result in inadequate inventory levels or unsold surplus merchandise. Hence, a proficient forecasting technique holds significant importance in accurately predicting the demand for rice at essential food establishments. This study aims to forecast the market for rice at basic food stalls to mitigate the issue of excessive inventory. Examining historical data about rice sales yields significant insights into the patterns and trends of demand. The data analysis enables the identification of fluctuations in the market that are likely to occur. This approach has the potential to enhance the precision of forecasting future demand.

The precise anticipation of rice demand in small-scale establishments primarily selling staple foods is crucial for optimizing inventory management and maintaining a steady rice supply. This study suggests the utilization of Neural Prophet, a robust model for time series forecasting, to predict rice sales at food stalls. The Neural Prophet model (Kılıç & Uğur, 2023) integrates the capabilities of neural networks and additive regression models, rendering them well-suited for identifying intricate patterns and trends in time series data. The objective of this study is to assess the efficacy of the Neural Prophet model in forecasting rice sales in fundamental food booths. The model was trained using historical sales data from primary food booths from June 2023 to April 2023. The model incorporates a range of elements, including seasonality, trends, and external variables, such as vacations, to enhance predictions' accuracy.

The researcher intends to perform empirical investigations by utilizing datasets obtained and compiled from sales logs recorded by an inventory application system or inventory system. The dataset was obtained over the period spanning from June 2023 to April 2023. The experiment's outcomes will yield insights into the model and its performance. The training procedure employed for the dataset involves utilizing the Deep Learning (Hindarto & Handri Santoso, 2021), (Hindarto & Santoso, 2022) technique known as Neural Prophet, specifically in the context of time series analysis. The description mentioned above prompts the formulation of a research question in this study: How can Neural Prophet training be conducted for time series datasets? Research Question 1. What is the level of performance exhibited by the model? Research Question 2.

LITERATURE REVIEW

Numerous scholarly investigations have extensively examined the topic of forecasting utilizing time series datasets. The subsequent passage delineates the algorithm employed. Specialized Application of Prophet in ICU Demand Forecasting. Although numerous studies have examined the effectiveness of the Prophet approach, a noticeable gap exists in its application to specialized domains such as ICU (Borges & Nascimento, 2022) demand forecasting during the COVID-19 pandemic. While research has explored the integration of the Prophet and LSTM models for this purpose, more comprehensive investigations are needed to understand this hybrid approach's unique challenges and benefits in accurately predicting ICU demand. Such studies would improve healthcare resource allocation and patient care during critical situations. Spatiotemporal Environmental Forecasting with Prophet and KNN (Zhang et al., 2022). While integrating the Prophet model with K-Nearest Neighbors (KNN) has demonstrated promising results for spatiotemporal environmental predictions like O₃ concentration, there needs to be more research on the broader applicability of such combined models for various environmental variables. Further research could investigate how the Prophet-KNN hybrid approach can enhance predictive accuracy for other pollutants, weather phenomena, or ecological factors. Mixed Models for Container Throughput Prediction (Munim et al., 2023). The literature recognizes the importance of accurate container throughput prediction for effective port planning and operational efficiency. However, a gap still needs to be in systematically comparing and evaluating mixed forecasting models, incorporating the Prophet approach, against individual methods for predicting container traffic volume. Investigating the synergistic effects of combining the Prophet model with other techniques could provide insights into optimizing container throughput forecasts. Forecasting Combinations and Enhanced Performance (ArunKumar et al., 2021). While studies have compared various time series methodologies, including ARIMA, SARIMA, HWES, and the Prophet model, the potential for improved performance through forecasting combinations still needs to be explored. Further research could delve into the intricacies of combining these methodologies, potentially revealing new strategies to enhance prediction accuracy across different types of time series data.

Prophet Model for Freight Rate Prediction:

Applying the Prophet model to forecast container freight rates offers a novel approach to address challenges in the shipping and logistics industry. Yet, there needs to be more comprehensive research investigating how the Prophet model, alongside other advanced forecasting techniques, can effectively manage risks associated with freight rate

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

fluctuations and supply chain expenses. Extended Forecasting with Prophet in Infectious Disease Modeling (Mao et al., 2018). The comparative analysis between the Prophet model and ARIMA for COVID-19 forecasting in Indonesia has shed light on their respective performance. However, a gap exists in understanding the factors contributing to the decreasing accuracy of the Prophet model over longer prediction horizons. Further exploring the limitations and potential enhancements for Prophet's performance in extended forecasting scenarios would enrich its applicability to infectious disease modeling. By addressing these gaps, researchers can enrich the understanding and applicability of the Prophet model, contributing to its continued development and effectiveness in diverse forecasting scenarios.

METHOD

Research Stages

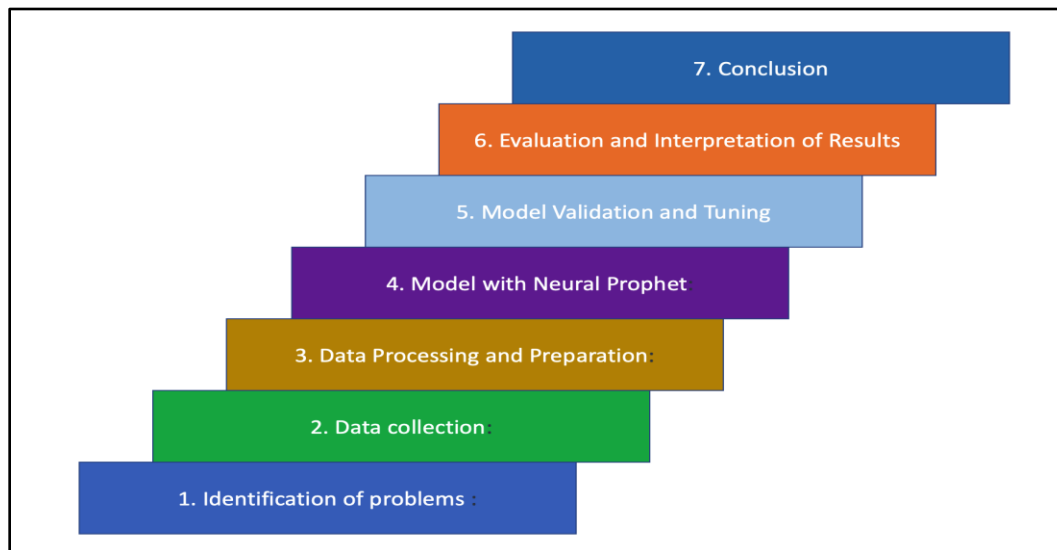


Fig 1. Step by step on Neural Prophet research
Source: Researcher Property

Fig 1 is a step by step in conducting research on times series datasets using Neural Prophet. Step 1, Problem identification is the process of clearly identifying and formulating the problem to be solved or the challenge that needs to be overcome. This involves an in-depth understanding of unwanted situations or conditions, barriers, unmet needs, or opportunities that could be improved. Effective problem identification is an important first step in designing the right solution. Understanding the context in which the problem arises is the first step in problem identification. This involves understanding the background, goals, and environment in which the problem arose. For example, in the context of "Rice Demand Prediction at Grocery Stalls," an understanding of the rice industry, staple food stalls, and the factors influencing rice demand will help identify relevant issues. Through direct observation, research, or data analysis, problem identification involves gathering information about the existing situation, obstacles encountered, or unmet needs. This can involve analyzing historical data, conducting interviews with relevant stakeholders, or observing existing processes. In the context of rice demand prediction, problem identification might involve understanding uncertainties in demand, difficulties in forecasting trends, or risks of overstock or understock.

Step 2, Data Collection is collecting the data needed for research. This could include historical data on the demand for rice at food stalls, such as the number of daily or weekly sales in the relevant timeframe. Variable identification is determining the variables to be measured and collected. In this case, the important variable is the demand for rice at the basic food stalls. Choose to collect daily or weekly sales data, depending on the desired level of detail. Specifies the data source to use. Collect data from food stalls directly, use available sales records, or work with related parties to gain access to relevant historical data. In addition, public data or data available at government agencies can also be a useful source of information. Determine the relevant timeframe for data collection. This timeframe should cover a period long enough to permit analysis of trends and demand patterns. The month and year selected must take into account seasonal variations and possible long-term trends. Make sure the data collected is of

* Corresponding author



good quality. Check the accuracy, completeness and consistency of the data. Clean data of missing, duplicate, or nonsensical values. If necessary, perform data pre-processing processes such as interpolation or exclusion of outliers to ensure better data quality. Storage of data in an appropriate and easily accessible format. Commonly used formats are spreadsheets such as Excel or CSV. Be sure to provide clear labels for each column of data to facilitate further analysis and data processing. It is important to keep the data collected secure. Be sure to follow applicable privacy and data protection policies, especially if the data includes customer information or other sensitive information.

Step 3, Data Processing and Preparation, the next step is to process and prepare it according to the format accepted by Neural Prophet. This may involve removing missing or incomplete data, performing data transformations where necessary, and dividing the data into a training set and a test set. Remove missing or incomplete data: Examine your dataset to identify if there are any missing or incomplete data. You can decide whether to delete rows or columns with missing data or use an appropriate method of filling in the data, such as filling using an average or interpolation. Data transformation: Some machine learning algorithms require data to be in a certain format or need to follow certain assumptions. Perform data transformations such as normalization (for example, using a 0-1 scale) or standardization (for example, changing the distribution to a normal distribution) to ensure that your data meets these requirements. Addressing outliers: Outliers can affect model performance. Choose to remove outliers if the data is invalid or perform special handling of these outliers, such as replacing them with valid values or using appropriate statistical methods. Separating dataset into training set and test set: It is important to divide dataset into training set and testing set. The training set is used to train the model, while the test set is used to test the performance of the model being trained. A common comparison is to divide the data by about 70-80% for the training set and 20-30% for the test set.

Step 4, Modeling with Neural Prophet, do data modeling using Neural Prophet. This involves importing the Neural Prophet library into the programming environment used. In this research using the Python programming language to define and initialize the model. Conduct model training using the prepared dataset. Import the Neural Prophet library into the Python programming environment. This is done using the import neural prophet command. After importing the Neural Prophet library, define and initialize the Neural Prophet model. Create a model object using the command `model = NeuralProphet()`. Set some relevant parameters, such as prediction interval, data frequency. Train the model using the prepared dataset. Uses the `fit()` method on the model object to train a model with a dataset. In the `fit()` method, defines a training dataset, selects an optimization method, and sets the appropriate training parameters. During the training process, the model will learn from the patterns in the dataset and optimize parameters to achieve better predictions.

Step 5, Model Validation and Tuning is to do model training. The purpose of this step is to perform validation to measure the performance of the model. Use of evaluation metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) to compare model predictions with actual values. If the model performance is not satisfactory, you can perform model tuning by changing the hyperparameters, such as adjusting the number of seasonal components or adjusting the level of confidence. Model Validation: After the model has been trained using the training dataset, the first step is to validate the model to measure its performance. Validation is done by comparing the model predictions with the actual values from the test dataset or validation dataset that are not used during training. Common evaluation metrics used for validation are the Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). MAE measures the average absolute error between predictions and actual values, while RMSE measures the average squared error between predictions and actual values. The lower the MAE or RMSE value, the better the model performance. Model Tuning: If the model performance is not satisfactory, the next step is to perform model tuning. Model tuning involves changing hyperparameters or model-parameters to improve model performance. Hyperparameters are parameters that are not generated by the model itself, but must be defined by the user. Some examples of hyperparameters in a model could include the number of seasonal components, level of confidence, or optimization method. In model tuning, experiment with different combinations of hyperparameter values. After that it checks the performance of the model after every try. The goal is to find hyperparameter combinations that provide better predictive results. Performance Evaluation After Tuning: After performing model tuning, the performance of the updated model should be re-evaluated using the same evaluation metrics as MAE or RMSE. This evaluation will provide an understanding of the extent to which hyperparameter tuning has improved model performance. If the model performance is satisfactory after tuning, continue using the model for further predictions. However, if performance is still inadequate, considering additional tuning steps or expanding the explored hyperparameter space.

Step 6, Evaluation and Interpretation of Results. After the model is applied and adjusted, it performs a final evaluation of the prediction results. Compare the predicted results with actual data and analyze errors if there are significant differences. In addition, interpreting the prediction results to understand trends, patterns or fluctuations

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

detected by the model. If there is a significant difference between the predicted results and the actual data, the next step is to analyze the error. Error analysis can involve mapping and understanding the causes of discrepancies between predictions and actual data. Patterns or trends missed by the model, external factors not accounted for in the model, or problems in data processing or preparation. This error analysis can provide insight into areas that need improvement in the model or data processing to improve performance. In addition to error evaluation, interpretation of prediction results is also important to understand trends, patterns or fluctuations detected by the model. Long-term trends, seasonal patterns, daily fluctuations, or significant changes in demand or behavior identified by the model. This interpretation understands the factors that affect demand or patterns of data that can be used for business decision making. The Neural Prophet model has been used to analyze causality or cause-and-effect relationships, this step involves interpreting causal findings in the context of the business or problem at hand. Understand the factors that significantly influence the demand for rice at basic food stalls and explain their impact on business.

Step 7. The conclusion is the final decision of the process series of all steps based on the experiments carried out. The success and limitations of the Neural Prophet method in predicting the demand for rice at basic food stalls.

The success of the Neural Prophet Method:

- If the Neural Prophet method provides accurate predictive results and matches the demand for rice at basic food stalls, it can be said that this method is successful in modeling and predicting trends or patterns of demand.
- If this method can provide better results compared to traditional or other prediction methods used previously, then the success of Neural Prophet can be considered significant.

Limitations of the Neural Prophet Method:

- Neural Prophet, as with every other prediction method, also has limitations. These limitations may include the inability to predict accurately when there are significant external factors affecting the demand for rice in basic food stalls. Examples are price changes, seasons, or special events.
- Other limitations may be related to the data used. If the available data is very limited, incomplete, or does not cover important factors, then the predictions produced by Neural Prophet may be inaccurate or irrelevant.

Implications of Research:

- The implication of this research is that Neural Prophet can be used as a useful tool in predicting the demand for rice in food stalls. If this method proves to be successful and provides adequate results, the implication is that basic food stall owners can rely on this method for more effective inventory planning and stock management.
- This research can also provide new insights about trends or patterns of demand for rice at basic food stalls and the factors that influence them. The implication is that this research can help grocery shop owners or related stakeholders better understand consumer behavior and make smarter decisions in managing their business.

Neural Prophet

The Neural Prophet method is an approach used to model and predict time series data using a neural network. This method was developed based on the Facebook Prophet architecture, but with an increase in the use of neural networks to overcome some of Prophet's limitations in dealing with complex data.

- **Basic Components:** The Neural Prophet method (Saeed et al., 2023) consists of two main components, namely the trend component and the seasonal component. The trend component is used to describe general trends or long-term patterns in data, while the seasonal component is used to describe patterns that repeat in data over time.
- **Neural Network:** This method uses a recursive neural network to model and predict time series data. This neural network allows the model to capture the temporal dependence between adjacent data points in a time series. In Neural Prophet, the neural network architecture used is the Long Short-Term Memory (LSTM) (Qiu et al., 2022), (Wijaya et al., 2023) network.
- **Regression Component:** In addition to the trend and seasonality components, the Neural Prophet method can also model the regression component. The regression component is used to include external factors that can influence the behavior of the time series data. If you have rice sales data and want to factor in the price of rice as a regressor, the Neural Prophet method can handle this.
- **Auto-tuning:** Neural Prophet's strength is its ability to auto-tune many of the hyperparameters associated with the model and training process. This method can automatically determine parameters such as the number of seasonal components, level of confidence, or the optimal optimization method based on the data provided.

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

- Handling of Missing Data: Neural Prophet has the ability to handle missing or incomplete data. This method can fill in missing values automatically using an approach called linear imputation, which takes advantage of the temporal properties of time series data.
- Interpretation of Results: Neural Prophet also provides the ability to interpret predictive results. You can see the trend and seasonality components found by the model, and thus, understand the trends or fluctuations detected in the data.

The Neural Prophet method provides a powerful framework for modeling and predicting time series data using more sophisticated neural network approaches. This allows the model to capture more complex patterns in the data and produce more accurate predictions.

The neural network architecture used in Neural Prophet is Long Short-Term Memory (LSTM), which is a type of Recursive Neural Network (Karthick et al., 2022). LSTMs are specifically designed to handle temporal dependencies in time series data. One of the main advantages of the LSTM is its ability to retain information over a longer period of time, enabling it to capture complex patterns and dependencies in time series. In Neural Prophet, LSTM is used to model and predict time series data. The LSTM architecture allows the model to remember previous information in the time series and use that information to predict future values. Thus, the LSTM can capture recurring patterns, seasonality, and long-term trends in data. In addition, LSTM also has a mechanism called a gate which allows it to regulate the flow of information in the network. This gate helps in controlling and regulating the extent to which new information must be considered and the extent to which old information must be remembered. This is especially useful in dealing with the problem of vanishing or exploding gradients that can occur in deep neural networks. By using the LSTM (Costa, 2022) architecture in Neural Prophet, models can learn and describe complex patterns in time series data, including seasonality, trends, and fluctuations. This enables Neural Prophet to produce accurate predictions and understand the characteristics of complex time series data.

Neural Prophet has strong capabilities in dealing with seasonal patterns in time series data. This method can automatically identify, model, and predict the seasonal component in data, including daily, weekly, monthly, yearly, and other seasonality. In Neural Prophet, the approach used to extract seasonal patterns from data is to use the Fourier transform. Fourier transform is a technique used to analyze signals or data in the frequency domain. In the context of Neural Prophet, the Fourier transform is used to decompose time series data into seasonal components with various frequencies. By performing a Fourier transform, Neural Prophet can identify seasonal patterns and frequencies in the data. Having derived the seasonal component from the Fourier transform, Neural Prophet models it separately. This model combines the seasonal component with the trend component and the regression component (if any) to produce the final prediction. In this process, Neural Prophet is able to adjust and handle seasonal fluctuations that may occur in the time series data. Neural Prophet also has the ability to identify irregular or irregular seasonality. This method can deal with seasonal patterns that do not follow a fixed or regular pattern. Thus, Neural Prophet can accommodate time series data that has complex and different seasonality. In practice, Neural Prophet will automatically find and adjust for the seasonal component of time series data. This means users do not need to manually identify and address seasonal patterns in data. Neural Prophet will automatically handle this process and produce predictions that accurately reflect seasonal patterns. With its ability to identify and model the seasonal component in time series data, Neural Prophet is a very effective tool in analyzing and predicting seasonal data, such as data with daily, weekly, monthly, yearly, and other patterns.

In Neural Prophet, the process of identifying long-term trends is carried out by separating the trend components from the time series data. The model will look for trend patterns in the data and estimate the appropriate trend. One of Neural Prophet's strengths is its ability to handle fluctuations or changes in trends. If trends in the data experience significant changes or irregular fluctuations, Neural Prophet will adjust the model to accommodate those changes. Neural Prophet can also model complex non-linear trends in data. This model is not only limited to linear trends, but can also capture more complex trend patterns such as exponential, sigmoidal, or polynomial trends. This provides flexibility in modeling the different types of trends that may occur in time series data. Neural Prophet combines a trend component with a seasonal component and a regression component (if any) in its modeling. Thus, the model can integrate long-term trend effects with seasonal components and external factors in the prediction process. Neural Prophet's ability to identify and model long-term trends, this model is a powerful tool for analyzing and predicting time series data that has significant trends

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

RESULT

Result

The predictions made on rice commodities in basic food stalls show the following. The dataset is taken from the transaction log in the inventory system. Raw data from transaction logs uses the MySQL database and was collected from June 2022 to April 2023. The raw data is then exported to an excel file. The data is disaggregated for the date and number of sales taken each day. Date feature and rice sales feature.

	ds	y
import pandas as pd	0	2022-06-07 15
from neuralprophet import NeuralProphet, set_log_level	1	2022-06-08 2
set_log_level("ERROR")	2	2022-06-09 4
import plotly.express as px	3	2022-06-10 1
import plotly.graph_objects as go	4	2022-06-11 4
	5	2022-06-12 1
df = pd.read_excel("Beras.xlsx")	6	2022-06-13 4
df.head(10)	7	2022-06-14 20
	8	2022-06-15 4
	9	2022-06-16 50

Fig 2. Source code and dataset view for prediction

The source code in Fig 2 imports the required modules, namely pandas, NeuralProphet, set_log_level from neuralprophet, as well as the plotly.express and plotly.graph_objects visualization modules. Next, the code reads data from an Excel file called "Beras.xlsx" and displays the first 10 rows of the read dataframe. It's worth noting that it reads the Excel file directly and ensures that the "Rice.xlsx" file is in the same directory as the Python script. After successfully reading the data into the df dataframe, proceed with the next steps in time series analysis or modeling using Neural Prophet.

```
m = NeuralProphet (
    growth = "linear",
    changepoints = None,
    n_changepoints=5,
    changepoints_range=0.95,
    trend_reg=0,
    trend_reg_threshold=False,
    yearly_seasonality="auto",
    weekly_seasonality="auto",
    daily_seasonality="auto",
    seasonality_mode="additive",
    seasonality_reg=0,
    n_lags=7,
    batch_size=32,
    learning_rate=None,
    epochs=100,
    loss_func="Huber",
    normalize="auto",
    impute_missing=True
)
df_train, df_test = m.split_df(df, valid_p = 0.2)

print("Dataset size:", len(df))
print("Train dataset size:", len(df_train))
```

* Corresponding author



```
print("Test dataset size:", len(df_test))
```

Output:

Dataset size: 328

Train dataset size: 264

Test dataset size: 71

Fig 3. Source code model Neural Prophet and split dataset 80:20

Explanation in Fig 3, Code initializes the Neural Prophet model with various configuration parameters. Next, the code splits the dataset into a training set and a test set using the `split_df` method provided by Neural Prophet. The parameters set in the initialization of the Neural Prophet model are as follows:

- `growth`: Specifies the growth type of the model, in this case "linear" is used.
- `changepoints`: List of changepoints in the dataset. In this example, no inflection point is specified.
- `n_changepoints`: Number of changepoints allowed. Here, it is set to 5.
- `changepoints_range`: The percentage of the data range used to find changepoints. Here, it is set to 0.95, which means it uses 95% of the data.
- `trend_reg`: Regression penalty for trend. In this example, it is set to 0.
- `trend_reg_threshold`: The threshold used to choose whether to apply the trend regression penalty or not.
- `yearly_seasonality`, `weekly_seasonality`, `daily_seasonality`: Determines whether to automatically use annual, weekly, and daily seasonality.
- `seasonality_mode`: The seasonal mode used, in this case "additive" is used.
- `seasonality_reg`: Regression penalty for seasonality.
- `n_lags`: Number of lags used in the model.
- `batch_size`: Size of batch used in training.
- `learning_rate`: The learning rate used in the training.
- `epochs`: Number of training iterations performed.
- `loss_func`: The loss function used in training, in this case "Huber" is used.
- `normalize`: Normalize the data, in this case "auto" is used.
- `impute_missing`: Fills in missing values in data by interpolation.

After initializing the model and parameter configuration, the code uses the `split_df` method to divide the dataset into a training set and a test set with a validation proportion of 0.2 (20%).

```
m = NeuralProphet()
metrics = m.fit(df, freq="D")
metrics.head(10)
```

Missing logger folder: /Users/djarothindarto/DeepLearning/NeuralProphet/Beras-2023/lightning_logs

Finding best initial lr: 100% 215/215 [01:16<00:00, 258.29it/s]

Epoch 291: 100% 291/291 [00:00<00:00, 3531.76it/s, loss=0.0309, v_num=0, MAE=12.90, RMSE=15.80, Loss=0.0313, RegLoss=0.000]

	MAE	RMSE	Loss	RegLoss	epoch
0	50.201336	58.638313	0.341909	0.0	0
1	48.938190	57.252522	0.330291	0.0	1
2	48.039513	56.195648	0.319587	0.0	2
3	47.418724	55.747444	0.312097	0.0	3
4	46.441658	54.909733	0.301345	0.0	4
5	45.660858	53.397785	0.293360	0.0	5
6	45.125675	53.152008	0.283458	0.0	6
7	43.175053	50.997059	0.263189	0.0	7
8	41.817287	49.469917	0.248839	0.0	8
9	39.507923	46.776085	0.226985	0.0	9

Fig 4. Source code training model Neural Prophet.

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Explanation of Fig 4, The Python script above uses Neural Prophet to train a time series forecasting model using the data contained in the DataFrame (df). In the first line, the NeuralProphet object, denoted by the variable 'm', is initialized. This creates an instantiation of the Neural Prophet model which will be used for the training process. Uses the fit() method to train the model. This method takes a DataFrame (df) as the main argument, which contains the time series data to be used for training. Another optional argument is 'freq', which specifies the frequency of the time series data, in this case set as "D" indicating daily data. The fit() method is executed, the Neural Prophet model will go through several iterations to train and adjust the model parameters to fit the data. This iteration involves model-parameter optimization using techniques such as backpropagation and stochastic gradient optimization. The results of model training are stored in the 'metrics' variable. These metrics provide information about the performance of the model during training, such as loss function, error metrics and other evaluation metrics. In this case, the last row of 'metrics' is used to display the first 10 rows of those metrics. Using the script above, we can train the Neural Prophet model with the time series data contained in the DataFrame (df) and get information about the performance of the model during training. This information can be used to understand the extent to which the model successfully adapted to the training data and give an idea of the potential performance of the model on data that has never been seen before.

In the Python script above, after the Neural Prophet model is initialized, the training process starts using the fit() method. The fit() method will perform iterative training on the Neural Prophet model to adjust the model parameters to match the given time series data. During the training process, Neural Prophet uses the stochastic gradient optimization method to minimize the loss function between the model's predicted value and the actual value of the data. This loss function describes how well the model estimates the observed data. During each iteration, the model parameters are adjusted in such a way that the loss function is minimized. In addition, Neural Prophet also uses the backpropagation technique to calculate the gradient of the loss function against the model parameters. This gradient is used to update parameter values through stochastic gradient optimization, gradually approaching optimal values. During training, the Neural Prophet model also calculates various evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and others. These metrics provide information about the performance of the model during training and make it possible to evaluate the extent to which the model has successfully adapted to the data. The results of the Neural Prophet model training are stored in the 'metrics' variable. This variable contains a DataFrame that contains model performance metrics at each training iteration. By displaying the first few rows of 'metrics', we can see how the model's performance has evolved over time and understand to what extent the model has converged or still requires further iteration to achieve optimal performance. By using this Python script, you can train the Neural Prophet model with time series data and monitor the performance of the model during the training process. This information can be used to determine whether the model is good enough or needs further adjustments before being used to make predictions on new data that has never been seen before.



Fig 5. This is a prediction using Neural Prophet

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Fig 5, In a Python script using a pre-trained Neural Prophet model (in an 'm' object) to create a predictive visualization using the plot() method. In the first line, use the plot() method with the 'forecast' argument. 'Forecast' is the prediction result generated by the Neural Prophet model based on the time series data used for training. The plot() method generates a prediction visualization that includes a graph of the original observed data as well as the model's predictions for the specified period. After the plot() method is executed, the visualization object (in the 'fig1' variable) is created. 'fig1' contains an image or graph that displays observational data and predictions from the Neural Prophet model. Finally, use the show() method on the visualization object ('fig1') to display the prediction graph. The show() method will display the image/graphic interactively or in a separate pop-up window, depending on the programming environment used. By using this script, you can easily visualize the predictions generated by the Neural Prophet model to see how far the model's predictions match the observational data. This helps in analyzing the quality of predictions, seeing identified trends or patterns, and gaining a better understanding of the model's performance in predicting time series data.

By running the script, a graph of the predicted results will be displayed. This graph will display two main components: actual observed data. Namely the time series data used in training. Predictions of the Neural Prophet model for the specified period. The graph will have an x-axis that shows the time. For example, in the format of a date or other time range and the y-axis showing values from time series data. The graph will show the observed data with dots or lines representing the actual observed values, while the model predictions will be shown with different lines or with shaded areas to show the range of predictions.

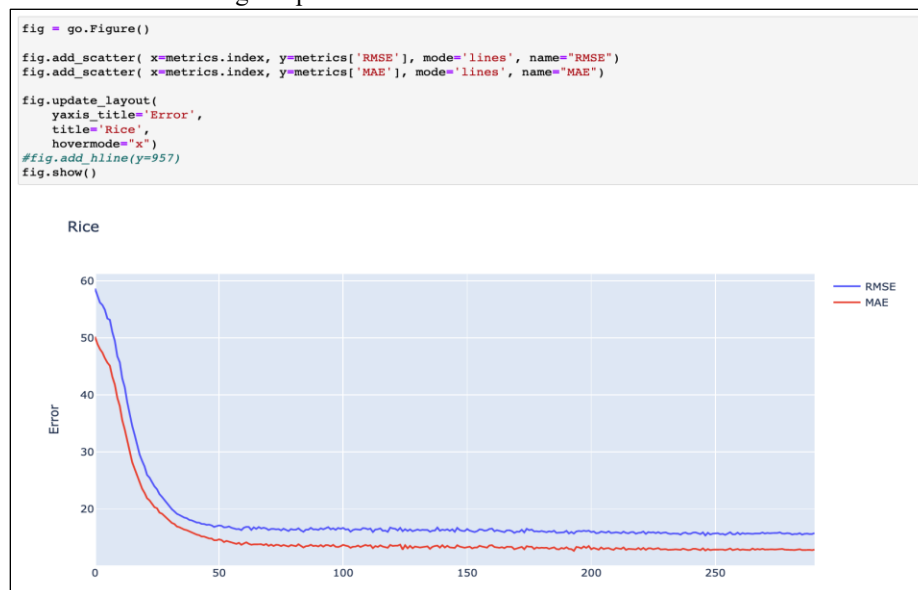


Figure 6.

The script in Figure 6, uses the Plotly library to create a graph using the data stored in the metrics DataFrame. The steps taken in the script are as follows:

- Create a fig object using go.Figure(). This object will be used as a container for the graphics to be created.
- Add line data using the add_scatter() method on the fig object. Two lines are added, namely 'RMSE' and 'MAE'. The data for the x-axis is taken from the DataFrame index metrics.index, while the data for the y-axis is taken from the 'RMSE' and 'MAE' columns in the DataFrame metrics. The 'lines' mode is used to indicate that lines will be drawn as straight lines. The line name is specified with the name parameter.
- Update the layout of the graph using the update_layout() method on the fig object. The y-axis label is specified with yaxis_title, the chart title is specified with title, and the hovermode is specified as "x" which means it only displays information on the x-axis on hover.
- Comment out the desired horizontal line by using add_hline() on the fig object and including the desired y value as a parameter. In the script above, the comments on the horizontal line are commented using a hash sign (#), so they are not executed.
- Displays a graph using the show() method on the fig object.

* Corresponding author



DISCUSSIONS

How to do Neural Prophet training for time series dataset? (RQ 1).

To conduct Neural Prophet training for time series datasets, you can follow these steps:

- **Data Preparation:** Make sure the time series data is ready for use and meets the requirements required by Neural Prophet. This includes making sure the data is sorted by time and consists of two columns: "ds" (date/time) and "y" (target value). Time Series Data, ensure that the time series data has been collected correctly and consists of two main columns: "ds" and "y". The "ds" field must contain date/time values in the proper format. Column "y" must contain the target value you want to predict, for example, the demand for rice at a basic food shop. Data Sorting, make sure data is sorted by column "ds" in correct time order. Neural Prophet requires sequenced data to produce accurate predictions. Adjustment of Data Format, If necessary, adjust the data format to suit Neural Prophet requirements. For example, ensure that the "y" column contains appropriate numeric values, and not strings or categorical values.
- **Package Installation and Import:** Ensure that you have installed the Neural Prophet package and import the required modules into the working environment. Use the pip command to install the Neural Prophet package, such as "pip install neuralprophet". Make sure you have Python installed in your working environment. Neural Prophet requires Python version 3.7 or later. Open a terminal or command prompt and run the following command to install the Neural Prophet package using pip.
- **Model Initialization:** Initialize the Neural Prophet model object by instantiating the NeuralProphet() class. Sets model-parameters such as desired number of seasons, resampling settings, or additional model options. Make sure to run the installation and import of the Neural Prophet package in the appropriate Python working environment. If there are conflicts or problems while installing or importing packages, make sure to have a compatible version of Python and check the official Neural Prophet documentation for more detailed installation guides or troubleshooting that may be platform related.
- **Data Division:** Divide the time series data into training sets and testing sets. The training set will be used to train the model, while the test set will be used to evaluate the performance of the model being trained. Define Split Proportion, define split proportion between training set and test set. In general, about 70-80% of the data can be used as a training set, while the remaining 20-30% can be used as a test set. However, this proportion can be adjusted based on the size of the dataset and specific needs. Time Order, make sure to maintain the time order when splitting the data. Data should be sorted by the "ds" (date/time) column in the correct order. Separate Data, use the data separation method according to the programming language or library used. For example, using Python and pandas, you can use the train_test_split() method to split the data into a training set and a test set.
- **Model Training:** Use the fit() method on the model object to train the model using a training set. For example, using the command "model.fit(df_train)" where df_train is a DataFrame containing training data with columns "ds" and "y".
- **Prediction:** After the model is trained, use the predict() method on the model object to make predictions on data that haven't been seen before.
For example, using the command "future=model.make_future_dataframe(periods=30)" to create a DataFrame with the time range you want to predict, then use "forecast = model.predict(future)" to get the prediction results.
- **Model Evaluation:** Evaluate the performance of the model by comparing predictions with actual values in the test set. Use of evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or coefficient of determination (R-squared) to measure model accuracy.
- **Results Visualization:** Visualize predicted results and compare them to actual data using appropriate graphs or plots. This will help understand to what extent the Neural Prophet model is capable of modeling and predicting patterns in time series data.
- **Model Setup:** If the evaluation results are not satisfactory, you can perform model tuning by changing the model parameters or changing the Neural Prophet configuration. Experiment with parameter variations, such as changing the number of seasons, adding additional components, or optimizing the model using the techniques provided by Neural Prophet.
- **Model Usage:** Once a model has been trained and evaluated, you can use it to make predictions on future data or data that has not been seen before. Be sure to pay attention to model assumptions and limitations, and track model performance periodically to ensure continuous accurate modeling.

By following the steps above, you can train a Neural Prophet for a time series dataset using relevant historical data. However, it should be noted that Neural Prophet has several parameters that can be adjusted for more advanced model

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

tuning, such as the number of seasonal components, the impact of holidays, and the use of additional features. In addition, Neural Prophet also has additional features such as missing value handling, weighting data, and the ability to evaluate models using different evaluation metrics. Explore the Neural Prophet documentation and use the provided usage examples to gain a deeper understanding of how to use and optimize the Neural Prophet model for time series datasets. During training, it is important to pay attention to general principles in model training, such as data separation, model validation, handling of overfitting and final evaluation. With a careful approach and setting the right parameters, it can produce an accurate and reliable Neural Prophet model to predict rice demand at basic food stalls or for other time series prediction cases.

How does the Neural Prophet model perform? (RQ 2).

The performance of the Neural Prophet model can be evaluated using various evaluation metrics that are commonly used to predict time series data. Some of the evaluation metrics that are often used are as follows:

- Mean Absolute Error (MAE): MAE measures the average of the absolute difference between the predicted value and the true value. The lower the MAE value, the better the model performance. MAE provides a measure of the average absolute error of the predictions.
- Mean Squared Error (MSE): MSE measures the average of the squared difference between the predicted value and the true value. The lower the MSE value, the better the model performance. MSE provides a measure of the mean squared error of the predictions.
- Root Mean Squared Error (RMSE): RMSE is the square root of the MSE and gives a measure of the average error between the predictions and the true value. The lower the RMSE value, the better the model performance.
- Mean Absolute Percentage Error (MAPE): MAPE measures the average absolute percentage error between the predicted value and the true value. The lower the MAPE value, the better the model performance. MAPE provides a measure of the average error in the form of a percentage of predictions.

In addition, used visualization to evaluate the performance of the Neural Prophet model. Plot graphs that compare the predicted values to the actual values can see to what extent the model is able to capture patterns and trends in the data. It should be remembered that the performance evaluation of the Neural Prophet model must be carried out not only on the training set, but also on the test set where data has not been seen before. This is important to ensure that the model can make good predictions on data that is not used in training. In addition to the standard evaluation metrics mentioned above, sometimes there are also special evaluation metrics that can be used depending on the needs and characteristics of the time series data being predicted. Therefore, it is always important to consider the context and specific goals of the time series prediction problem encountered when evaluating the performance of the Neural Prophet model.

CONCLUSION

In the basic food stall industry, predicting stock and demand for products such as rice is very important to optimize inventory and avoid shortages or oversupply. In an effort to improve prediction accuracy, the Neural Prophet Time Series is one method that can be used. This method combines the advantages of neural network algorithms and time series models to produce more accurate and reliable predictions. In implementing the Neural Prophet Time Series to predict rice in basic food stalls, the first step is to prepare historical data that includes rice stock and demand. This data can then be split into training data and validation data to test model performance. Next, the model training process is carried out using the fit() method on the Neural Prophet object. The training process involves iterating to adjust the model parameters to suit the training data. During training, evaluation metrics such as MAE, MSE, and RMSE are used to measure the quality of the model's predictions. After the model is trained, the next step is to generate predictions using the plot() method. This method will produce a graph that displays the actual rice observation data as well as the predictions of the Neural Prophet model for the specified period. With this visualization, the owner of a basic food shop can easily see how far the model's predictions match the observational data. This makes it possible to make more informed decisions in managing the supply and demand for rice at basic food stalls.

The use of the Neural Prophet Time Series in predicting rice in basic food stalls can provide several advantages. First, this method combines the strengths of neural networks and time series models, so that they can capture complex patterns and trends in historical data. This allows for more accurate predictions and is responsive to fluctuations in market demand. Second, by using evaluation metrics such as MAE, MSE, and RMSE, basic food stall owners can measure and monitor the quality of model predictions during training. This allowed them to optimize the model-parameters and improve the quality of the predictions over time. With more accurate predictions, basic food shop

* Corresponding author



owners can manage rice supplies more efficiently. They can avoid stock shortages which can lead to lost customers, as well as reduce oversupply which can result in financial losses. In addition, by knowing the anticipated demand, the owner of the basic food stall can plan a more effective promotion or sales strategy. Overall, the use of the Neural Prophet Time Series in predicting rice in basic food stalls can be an invaluable tool for increasing operational efficiency and business profits for food stalls. With accurate predictions, basic food stall owners can avoid stock shortages which can result in losing customers and harming business image. In addition, by knowing anticipated demand trends, basic food stall owners can optimize procurement strategies and price adjustments to maximize profits.

REFERENCES

- ArunKumar, K. E., Kalaga, D. V., Sai Kumar, C. M., Chilkoor, G., Kawaji, M., & Brenza, T. M. (2021). Forecasting the dynamics of cumulative COVID-19 cases (confirmed, recovered and deaths) for top-16 countries using statistical machine learning models: Auto-Regressive Integrated Moving Average (ARIMA) and Seasonal Auto-Regressive Integrated Moving Averag. *Applied Soft Computing*, 103(December 2019), 107161. <https://doi.org/10.1016/j.asoc.2021.107161>
- Borges, D., & Nascimento, M. C. V. (2022). COVID-19 ICU demand forecasting: A two-stage Prophet-LSTM approach. *Applied Soft Computing*, 125, 109181. <https://doi.org/10.1016/j.asoc.2022.109181>
- Costa, R. L. de C. (2022). Convolutional-LSTM networks and generalization in forecasting of household photovoltaic generation. *Engineering Applications of Artificial Intelligence*, 116(September), 105458. <https://doi.org/10.1016/j.engappai.2022.105458>
- Hindarto, D., & Handri Santoso. (2021). Android APK Identification using Non Neural Network and Neural Network Classifier. *Journal of Computer Science and Informatics Engineering (J-Cosine)*, 5(2), 149–157. <https://doi.org/10.29303/jcosine.v5i2.420>
- Hindarto, D., & Santoso, H. (2022). PERFORMANCE COMPARISON OF SUPERVISED LEARNING USING NON-NEURAL NETWORK AND NEURAL NETWORK. *Janapati*, 11, 49–62.
- Karthick, M. K., Kiruthiga, G., Ms Saraswathi, P., Dhiyanesh, B., & Radha, R. (2022). A Subset Scaling Recursive Feature Collection Based DDoS Detection Using Behavioural Based Ideal Neural Network For Security In A Cloud Environment. *Procedia Computer Science*, 215, 509–518. <https://doi.org/10.1016/j.procs.2022.12.053>
- Kılıç, D. K., & Uğur, Ö. (2023). Hybrid wavelet-neural network models for time series. *Applied Soft Computing*, 144, 110469. <https://doi.org/10.1016/j.asoc.2023.110469>
- Mao, Q., Zhang, K., Yan, W., & Cheng, C. (2018). Forecasting the incidence of tuberculosis in China using the seasonal auto-regressive integrated moving average (SARIMA) model. *Journal of Infection and Public Health*, 11(5), 707–712. <https://doi.org/10.1016/j.jiph.2018.04.009>
- Munim, Z. H., Fiskin, C. S., Nepal, B., & Chowdhury, M. M. H. (2023). Forecasting container throughput of major Asian ports using the Prophet and hybrid time series models. *Asian Journal of Shipping and Logistics*, xxxx, 1–11. <https://doi.org/10.1016/j.ajsl.2023.02.004>
- Qiu, K., Li, J., & Chen, D. (2022). Optimized long short-term memory (LSTM) network for performance prediction in unconventional reservoirs. *Energy Reports*, 8, 15436–15445. <https://doi.org/10.1016/j.egy.2022.11.130>
- Saeed, N., Nguyen, S., Cullinane, K., Gekara, V., & Chhetri, P. (2023). Forecasting container freight rates using the Prophet forecasting method. *Transport Policy*, 133(December 2022), 86–107. <https://doi.org/10.1016/j.tranpol.2023.01.012>
- Saputra, A. D., Hindarto, D., & Santoso, H. (2023). Disease Classification on Rice Leaves using DenseNet121, DenseNet169, DenseNet201. *Sinkron*, 8(1), 48–55. <https://doi.org/10.33395/sinkron.v8i1.11906>
- Wijaya, S., Heryadi, Y., Arifin, Y., Suparta, W., & Lukas. (2023). Long short-term memory (LSTM) model-based reinforcement learning for nonlinear mass spring damper system control. *Procedia Computer Science*, 216(2022), 213–220. <https://doi.org/10.1016/j.procs.2022.12.129>
- Wu, F., Wang, Y., Liu, Y., Liu, Y., & Zhang, Y. (2021). Simulated responses of global rice trade to variations in yield under climate change: Evidence from main rice-producing countries. *Journal of Cleaner Production*, 281, 124690. <https://doi.org/10.1016/J.JCLEPRO.2020.124690>
- Zhang, B., Song, C., Li, Y., & Jiang, X. (2022). Spatiotemporal prediction of O3 concentration based on the KNN-Prophet-LSTM model. *Heliyon*, 8(11), e11670. <https://doi.org/10.1016/j.heliyon.2022.e11670>
- ZHOU, Q., YUAN, R., ZHANG, W. yang, GU, J. fei, LIU, L. jun, ZHANG, H., WANG, Z. qin, & YANG, J. chang. (2023). Grain yield, nitrogen use efficiency and physiological performance of indica/japonica hybrid rice in response to various nitrogen rates. *Journal of Integrative Agriculture*, 22(1), 63–79.

* Corresponding author



<https://doi.org/10.1016/j.jia.2022.08.076>

