FORECASTING MALAYSIAN RICE PRODUCTION USING HISTORICAL CLIMATE DATA AND MACHINE LEARNING ALGORITHMS

NURHAFIZAH BINTI MOHD YUNOS

UNIVERSITI TEKNOLOGI MALAYSIA

TABLE OF CONTENTS

TITLE **PAGE**

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Research Design

This study adopts a quantitative research design with a focus on predictive modelling. The central objective is to forecast Malaysia's rice (paddy) production using historical climate data and machine learning techniques. By implementing and comparing three different regression-based models—Random Forest, Support Vector Regression (SVR), and Long Short-Term Memory (LSTM)—the research aims to determine the most effective algorithm for accurate yield forecasting under varying climatic conditions.

The methodological framework includes several key stages:

a) Data collection and preprocessing

b) Exploratory Data Analysis (EDA)

c) Feature engineering

d) Model development and training

e) Model evaluation and comparison

Python was the primary programming language used throughout the study, supported by various libraries such as pandas, numpy, scikit-learn, matplotlib, seaborn, and TensorFlow/Keras.

3.2 Data Sources

3.2.1 Crop Production Dataset

Historical paddy production data were sourced from the Department of Statistics Malaysia (DOSM). This dataset includes monthly production statistics across multiple states from 2017 to 2022. The key attributes are:

a) State: Name of the state

b) date: Observation date in year-month format

c) crop_type: Crop category (paddy)

d) planted_area: Total cultivated area in hectares

e) Production: Total rice output in metric tons

3.2.2 Climate Dataset

Climate-related variables were retrieved from NASA's POWER (Prediction Of Worldwide Energy Resource) project. These variables were extracted monthly for each state and merged with crop data. The primary climate variables used include:

a) PRECTOTCORR_SUM: Total monthly precipitation (mm)

b) T2M_MAX: Monthly maximum temperature (deg C)

c) T2M_MIN: Monthly minimum temperature (deg C)

d) RH2M: Relative humidity at 2 meters (%)

e) ALLSKY_SFC_LW_DWN: Surface longwave radiation (W/m^2)

3.3 Data Description

The final merged dataset consisted of 792 rows and 20 columns, with each row representing a monthly observation per state. Important variables included:

a) State, date, planted_area, production

b) Climate variables, as mentioned above

c) yield: Calculated as production / planted_area

d) Time-based variables: month, year, week_of_year

e) Lagged variables: e.g., production_lag_1, yield_lag_1, production_lag_2, etc.

3.4 Data Preprocessing

Data preprocessing is a critical step in any machine learning project as it ensures that the data is clean, consistent, and ready for model training. In this study, several preprocessing steps were applied to both the crop production dataset obtained from the Department of Statistics Malaysia (DOSM) and the climate dataset sourced from NASA's POWER project.

3.4.1 Data Cleaning

The initial step involved cleaning the datasets to remove inconsistencies and ensure uniformity:

a) The column name 'NAME' was renamed to 'state' for clarity.

b) All state names were standardised to lowercase to avoid case sensitivity issues.

c) The 'date' column was converted into a proper datetime format to facilitate time-based operations such as sorting and feature extraction.

d) Missing values were checked using .isnull().sum(), and no missing values were found in either dataset.

e) Duplicate rows were identified using .duplicated().sum(), and none were present.

3.4.2 Dummy Data Generation

Because the crop data was originally provided as annual totals, a monthly dummy allocation was implemented based on a seasonal distribution pattern, enabling time-aligned comparison with monthly climate data.

3.4.3 Merging Datasets

To integrate climate variables into the analysis, the monthly paddy dataset was merged with the processed climate dataset using the keys 'state' and 'date'. An inner join was performed to retain only those records where both paddy and climate data were available. This resulted in a final merged dataset containing 792 rows and 20 columns. An inner join

was used to merge the climate and production datasets on state and date, ensuring only matched records were retained.

3.4.4 Feature Engineering

Several new features were introduced to improve model performance:

- Target Variable ('yield'): Calculated as the ratio of production to planted area (production / planted_area). To handle potential division by zero, infinite values were replaced with zero.

- Lagged Features: Lagged versions of production and yield were created to capture temporal dependencies:
  - production_lag_1, production_lag_2, production_lag_3
  - yield_lag_1, yield_lag_2, yield_lag_3
  - These lagged features were generated using groupby('state') to maintain consistency within each state's time series.

- Time-Based Features: Additional time-related features were extracted from the 'date' column:
  - month: Extracted as an integer (1–12)
  - year: Extracted as an integer (e.g., 2017, 2018)
  - week_of_year: Derived using .dt.isocalendar().week

- These engineered features enhanced the models' ability to capture seasonality and trends over time.

3.4.5 Train-Test Split

To preserve the temporal order of observations in the time series, a chronological train-test split was applied:

a) Training Set: 80% of the data (607 samples), covering earlier periods

b) Testing Set: 20% of the data (152 samples), consisting of the most recent observations

This ensured that the model was evaluated on its ability to predict future values rather than past or random ones.

3.4.6 Feature Scaling

Since Support Vector Regression (SVR) and Long Short-Term Memory (LSTM) are sensitive to the scale of input features, all numerical features were standardised using StandardScaler:

a) Each feature was transformed to have a zero mean and unit variance.

b) The target variable (production) was also scaled before model training and later inverse-transformed after prediction to return to the original units.

All preprocessing steps were carried out using Python libraries such as pandas, numpy, and scikit-learn.

3.5 Machine Learning Models

This section outlines the three machine learning models used in this study to forecast Malaysian rice (paddy) production using historical climate data. These models include Random Forest Regressor, Support Vector Regression (SVR), and Long Short-Term Memory (LSTM) neural networks. Each model was selected based on its suitability for regression tasks and its ability to capture complex patterns in time-series data.

3.5.1 Random Forest Regressor

The Random Forest Regressor is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It works by training each tree on a random subset of the data and features, then averaging the predictions across all trees.

a) Advantages:

- Handles non-linearities
- Resistant to overfitting
- Provides feature importance metrics

b) Hyperparameters:

- n_estimators = 100
- random_state = 42
- n_jobs = -1 (parallel processing)

The model was trained on scaled input features and evaluated using standard regression metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2).

3.5.2 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a kernel-based algorithm that performs regression by finding a hyperplane that best fits the data within a certain margin of tolerance (epsilon). It is particularly effective in high-dimensional spaces and can handle small datasets with good generalisation performance.

a) Advantages:

- Performs well with high-dimensional and small datasets

b) Hyperparameters:

- Kernel: rbf
- C = 100, gamma = 0.1, epsilon = 0.1

Before training, both input features and target variables were scaled using StandardScaler due to SVR's sensitivity to feature scales.

3.5.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to learn long-term dependencies in sequential data. It is widely used for time-series forecasting because it can capture temporal patterns and trends effectively.

a) Architecture:

- One LSTM layer with 50 units
- Dropout layer (rate = 0.2)
- Dense output layer

b) Training Parameters:

- Optimiser: Adam
- Loss function: MSE
- Early stopping with patience = 10
- Batch size: 32, Epochs: 100

Input data was reshaped into a 3D format [samples, timesteps, features] required by LSTM. The target variable was also scaled before training and inverse-transformed after prediction for interpretation.

3.6 Model Development Process

Step 1: Define Input Features and Target Variable

The first step involved selecting the most relevant input features and defining the target variable for prediction.

a) Input Features (X):

- planted_area
- PRECTOTCORR_SUM (Total monthly precipitation)
- T2M_MAX (Maximum temperature)

b) Lagged features:

- production_lag_1, production_lag_2, production_lag_3
- yield_lag_1, yield_lag_2, yield_lag_3

c) Time-based features:

- month, year, week_of_year

d) Target Variable (y):

- Production: Total paddy production in metric tons

These features were selected based on their relevance to crop yield and their ability to capture temporal patterns through lagging and time-based engineering.

Step 2: Train-Test Data Splitting

To ensure that the model was evaluated on its ability to predict future values rather than past or random ones, a chronological train-test split was applied:

a) Training Set: 80% of the data (607 samples), covering earlier periods

b) Testing Set: 20% of the data (152 samples), consisting of the most recent observations

This split preserved the order of time-series data and ensured a realistic evaluation of forecasting performance.

Step 3: Feature Scaling

Since Support Vector Regression (SVR) and Long Short-Term Memory (LSTM) are sensitive to feature scales, all numerical features were standardised using StandardScaler from scikit-learn.

a) Each feature was transformed to have a zero mean and unit variance.

b) The target variable (production) was also scaled before model training and later inverse-transformed after prediction to return to the original units.

Step 4: Model Training

Random Forest Regressor, Support Vector Regression (SVR), and Long Short-Term Memory (LSTM) were trained using the preprocessed dataset. The dataset was split into training and testing sets, with 80% of the data used for training and the remaining 20% reserved for evaluation. To ensure consistency across models sensitive to feature scales, such as SVR and LSTM, all numerical input features were standardised using StandardScaler,

transforming them to have zero mean and unit variance. The target variable (production) was also scaled before training and later inverse-transformed after prediction for interpretability.

For the Random Forest Regressor, a tree-based ensemble method, the model was initialised with 100 decision trees (n_estimators=100) and trained on the scaled training data. Random Forest does not require strict feature scaling, but it was applied for consistency across models. The model learned patterns from the input features, including lagged production values, time-based features, and climate variables, to predict paddy production.

The Support Vector Regression (SVR) model was trained using a Radial Basis Function (RBF) kernel, which is effective in capturing non-linear relationships in high-dimensional spaces. Hyperparameters such as regularisation (C=100), kernel coefficient (gamma=0.1), and tube size (epsilon=0.1) were set based on prior experimentation. Due to SVR's sensitivity to input scale, both the input features and target variable were scaled before training.

Finally, the LSTM model, a type of recurrent neural network suitable for sequence prediction tasks, was built using TensorFlow/Keras. Input data was reshaped into a 3D format [samples, timesteps, features] required by LSTM. The architecture included one LSTM layer with 50 units, followed by a dropout layer to prevent overfitting, and a dense output layer. The model was compiled using the Adam optimiser and Mean Squared Error (MSE) loss function. Early stopping was implemented during training to halt the process if validation loss

did not improve for 10 consecutive epochs, preventing overfitting and saving computation time.

Step 5: Prediction and Evaluation

After training, each model made predictions on the test dataset. Predicted values were inverse-scaled to their original units before evaluation. The evaluation metrics used: Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared (R^2 Score)

The model development process included defining input-output variables, splitting data chronologically, applying feature scaling, training each model, and evaluating their performance using standard metrics.

3.7 Evaluation Metrics

To evaluate the accuracy and effectiveness of the developed machine learning models in forecasting paddy production, three widely accepted regression evaluation metrics were used.

3.7.1 Mean Squared Error (MSE)

The Mean Squared Error (MSE) measures the average squared difference between the actual values and the predicted values. It emphasises larger errors due to the squaring operation, making it sensitive to outliers.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

3.7.2 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is the square root of the MSE. It provides an error metric in the same unit as the target variable (paddy production), making it more interpretable than MSE.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3.7.3 R-squared (R^2)

The R-squared (R^2) score measures how well the model explains the variability of the target variable. It ranges from 0 to 1, where:

- 1 indicates a perfect fit
- 0 indicates that the model performs no better than predicting the mean of the target

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

3.8 Tools and Technologies

On the Insert tab, the galleries include items that are designed to coordinate with the theme

Component	Description
Programming Language	Python 3.11

Libraries	pandas, numpy, scikit-learn, matplotlib, seaborn, TensorFlow/Keras
Platform	Google Colab Pro (with GPU support)

3.9 Summary

This chapter provided a detailed overview of the research methodology employed to forecast rice production in Malaysia. It described the data sources, preprocessing techniques, feature engineering methods, and the architecture of the three machine learning models. The evaluation metrics and tools used were also discussed. The next chapter presents the results of the model training process and provides a comparative performance analysis across the three algorithms.