

UNIVERSITY OF THE PHILIPPINES MANILA
COLLEGE OF ARTS AND SCIENCES
DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS

FORECASTING THE QUARTERLY PRODUCTION OF
RICE AND CORN IN DAVAO DEL SUR: A TIME
SERIES ANALYSIS

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Cedric M. Caquilala

July 2023

Permission is given for the following people to have access to this SP:

| | |
|--|-----|
| Available to the general public | Yes |
| Available only after consultation with author/SP adviser | No |
| Available only to those bound by confidentiality agreement | No |

ACCEPTANCE SHEET

The Special Problem entitled "Forecasting The Quarterly Production Of Rice And Corn In Davao del Sur: A Time Series Analysis" prepared and submitted by Cedric M. Caquilala in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Perlita E. Gasmen, M.Sc.(*cand.*)
Adviser

Alex C. Gonzaga, Ph.D.
Co-Adviser

EXAMINERS:

| | Approved | Disapproved |
|--|----------|-------------|
| 1. Avegail D. Carpio, M.Sc. | _____ | _____ |
| 2. Richard Bryann L. Chua, M.Sc.(<i>cand.</i>) | _____ | _____ |
| 3. Ma. Sheila A. Magboo, Ph.D. (<i>cand.</i>) | _____ | _____ |
| 4. Vincent Peter C. Magboo, M.D. | _____ | _____ |
| 5. Marbert John C. Marasigan, M.Sc. (<i>cand.</i>) | _____ | _____ |
| 6. Geoffrey A. Solano, Ph.D. | _____ | _____ |

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

| | |
|--|---|
| Vio Jianu C. Mojica, M.Sc. Unit Head Mathematical and Computing Sciences Unit Department of Physical Sciences and Mathematics | Marie Josephine M. De Luna, Ph.D. Chair Department of Physical Sciences and Mathematics |
|--|---|

Maria Constancia O. Carrillo, Ph.D.
Dean
College of Arts and Sciences

Abstract

Time Series Analysis is a valuable tool in making informed decisions. The Philippines would greatly benefit in using this for crop production, especially rice and corn, since the country is a large producer and consumer said products. The study aims to develop a system that predicts rice and corn production using datasets from Davao del Sur using SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM. The obtained models provide, at the lowest, an 8.42% MAPE for rice and 19.87% MAPE for corn. In addition, the system developed allows the user to develop their own models.

Keywords: Time series analysis, SARIMA, Bayesian SARIMA, Holt-Winters, LSTM

Contents

| | |
|---|-------------|
| Acceptance Sheet | i |
| Abstract | ii |
| List of Figures | vii |
| List of Tables | viii |
| I. Introduction | 1 |
| A. Background of the Study | 1 |
| B. Statement of the Problem | 2 |
| C. Objectives of the Study | 3 |
| D. Significance of the Project | 4 |
| E. Scope and Limitations | 4 |
| F. Assumptions | 5 |
| II. Review of Related Literature | 6 |
| A. Related studies involving ARIMA | 6 |
| 1. Empirical analysis for crop yield forecasting in India | 6 |
| B. Related studies involving SARIMA | 7 |
| 1. ARIMA model for forecasting of rice production in India by using SAS | 7 |
| 2. Forecasting on the crude palm oil production in Malaysia using SARIMA Model | 8 |
| C. Related studies involving ARMA | 8 |
| 1. A guide to solar power forecasting using ARMA models | 8 |
| D. Related studies involving Holt-Winters | 9 |
| 1. Forecasting of onion prices in Bangalore market: an appli- cation of time series models | 9 |

| | | |
|-------------|---|-----------|
| 2. | Modeling and forecasting number of confirmed and death caused COVID-19 in IRAN: A comparison of time series forecasting methods | 10 |
| 3. | Long-Term Forecasting of Electrical Loads in Kuwait Using Prophet and Holt–Winters Models | 10 |
| E. | Related studies involving LSTM | 11 |
| 1. | Forecasting of rice cultivation in India - a comparative analysis with ARIMA and LSTM-NN models | 11 |
| 2. | A Comparison of ARIMA and LSTM in Forecasting Time Series | 11 |
| 3. | LSTM Neural Network Based Forecasting Model for Wheat Production in Pakistan | 12 |
| F. | Conclusion | 13 |
| III. | Theoretical Framework | 14 |
| A. | Time Series Analysis | 14 |
| B. | Autoregressive Moving Average (ARMA) Model | 14 |
| C. | Autoregressive Integrated Moving Average (ARIMA) Model | 15 |
| D. | Seasonal ARIMA (SARIMA) Model | 16 |
| E. | Bayesian SARIMA Model | 16 |
| F. | Holt-Winters Exponential Smoothing | 17 |
| G. | Long Short-Term Memory Model | 17 |
| H. | Accuracy Metrics | 19 |
| I. | Existing Libraries | 20 |
| IV. | Design and Implementation | 21 |
| A. | Use Cases | 21 |
| B. | The Dataset | 22 |
| C. | Experimental Setup | 22 |
| 1. | SARIMA | 24 |

| | | |
|--------------|--|-----------|
| 2. | Bayesian SARIMA | 24 |
| 3. | Long Short-Term Memory | 25 |
| 4. | Holt-Winters Exponential Smoothing | 25 |
| D. | System Architecture | 26 |
| E. | Technical Architecture | 26 |
| V. | Results | 28 |
| A. | Landing Page | 28 |
| B. | Model Page | 28 |
| C. | Dataset Page | 32 |
| VI. | Discussions | 33 |
| A. | Dataset | 33 |
| B. | Model Development | 33 |
| 1. | Accuracy Metrics | 34 |
| 2. | Forecasts | 36 |
| C. | Discussion | 36 |
| VII. | Conclusions | 39 |
| VIII. | Recommendations | 40 |
| IX. | Bibliography | 41 |
| X. | Appendix | 44 |
| A. | Source Code | 44 |
| 1. | Imports | 44 |
| 2. | General Utils | 44 |
| 3. | SARIMA Source Code | 46 |
| 4. | Bayesian SARIMA Source Code | 47 |
| 5. | Holt-Winters Source Code | 48 |
| 6. | LSTM Source Code | 49 |

List of Figures

| | | |
|----|--|----|
| 1 | A diagram representing an LSTM Cell [1] | 18 |
| 2 | Use Case Diagram | 21 |
| 3 | Diagram of the experimental setup based on Haider[2] | 23 |
| 4 | Landing Page | 28 |
| 5 | Graphical summary of all models | 29 |
| 6 | Tabulized summary of all models | 29 |
| 7 | Summary of a SARIMA model | 30 |
| 8 | Forecasted values of a SARIMA model | 31 |
| 9 | Add model popup | 31 |
| 10 | Edit dataset page | 32 |
| 11 | Edit dataset cont. | 32 |
| 12 | Actual and fitted values of rice for each model | 35 |
| 13 | Actual and fitted values of corn for each model | 35 |

List of Tables

| | | |
|---|---|----|
| 1 | Accuracy metrics for rice | 34 |
| 2 | Accuracy metrics for corn | 36 |
| 3 | Forecasts for rice until 2025 | 37 |
| 4 | Forecasts for corn until 2025 | 37 |

I. Introduction

A. Background of the Study

The Philippines is a heavy consumer of rice and corn. The Department of Science and Technology – Food and Nutrition Research Institute (DOST-FNRI) are responsible for the examination of good consumption of the people. Based on the Expanded National Nutrition Survey by the DOST-FNRI, the total meals an average family consumes in a day is composed of 39% of cereal and cereal products [3]. Thus, we can safely conclude that rice and corn are both a major part of the daily nourishment required by the Filipino.

On the other hand, as much as the Philippines consumes rice and corn, it is also a significant producer of said products around the world. According to the Food and Agricultural Organization of the United Nations, the Philippines is the 8th top producer of rice since 2016. In addition, the Philippines is among the top 20 countries which produced the highest amount of corn in 2020 [4].

There is no doubt that the Philippines is both a massive producer and consumer of rice and corn. Yet despite this, the Philippines is still a heavy importer of rice, amounting to up to 2.01 million tonnes of imports from overseas in 2020 alone [5]. This number does not even consider the effects of unforeseen events such as natural disasters and international conflicts. According to the United States Department of Agriculture, this import is expected to further rise to 3.4 million tonnes after numerous factors such as rising fertilizer costs, the war on Ukraine, as well as the recent typhoon Karding [6].

Because of the importance of rice and corn in Philippine nutrition, it is vital that extensive research should be conducted to properly manage its supplies. One such way is predicting the possible production rate of the products. There have been existing studies with this purpose. A few examples would be a study on rice prediction by G. Ramakrishna and R. V. Kumari in 2018 [7], and another study which predicts a different crop by S. Dharmaraja et al. in 2020 [8]. Both

studies have been successful in developing models for their respective subjects. However, in the local context, there have been little to no research regarding the topic despite its relevance. While there have been forecasting models that have been created in other countries, the results may not always be applicable to the Philippines. Thus, it is essential to create a model based on datasets in the Philippines. One of the candidate locations for a preliminary analysis of the rice and corn production in the Philippines is Davao del Sur. According to OpenStat by the Philippine Statistics Authority (PSA), Davao del Sur accounts for about 35% of the production of rice in the Davao Region. This makes it an optimal sample that can represent the region in question.

There are many existing methods to create the described model. Among them are the Autoregressive Integrated Moving Average (ARIMA) and Autoregressive Moving Average (ARMA) models, which are the most commonly used methodology in dealing with univariate time series analysis. In addition, numerous other approaches such as Exponential Smoothing (ES) and Long Short-Term Memory (LSTM) are being more prevalent with more recent studies. Some of these studies result in these methods outperforming the traditional time series analysis models such as ARIMA. Examples of these are the rice cultivation study by K. K. Paidipati & Arjun Banik in 2019 [9] and the commodity price forecasting study by S. Siami-Namini et al. 2018 [10]. However, these are not conclusive for all subjects. Thus, using multiple methods are necessary for the special problem. These methods have existing foreign studies where they were implemented to make predictions for production rates of other products to much success. Similarly, these methods are the most applicable for the analysis of rice and corn production rates in the Philippines.

B. Statement of the Problem

Despite being a significant producer of rice and corn, the Philippines currently has no accessible tools to forecast the production of both crops in the Philippines [4].

Some studies have been conducted in relation to the crops in other countries, but different factors may make these tools inapplicable in the country. By developing a model using local datasets, a forecasting system can be designed that thoroughly takes into account the context in the Philippines.

C. Objectives of the Study

The study aims to develop a tool that can predict upcoming rice and corn production. The tool has the following functionalities:

1. The tool receives the dataset and saves it into the system.
2. It develops different models from methodologies such as SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM methods.
3. Time series graphs are shown which includes the plots of the dataset, the predicted model, and the resulting forecast for future years. All models used display a graph to make it easier to compare.
4. The characteristics of all models are also be displayed including accuracy metrics such as the Mean Absolute Deviation (MAD), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE).

On the other hand, the user is able to do the following:

1. The user is able to view all the forecasts of all models: SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM.
2. The user is able to choose the end date of the predictions.
3. The user is able to add, edit, or remove records to the time series.
4. The user is able to select their own parameters for the different models. For example, they can choose the parameters of $SARIMA(p, d, q)(P, D, Q)m$ to the model they create.

5. The user is able to save and load their models into the system.

D. Significance of the Project

- Producers. The result of this study is helpful to the producers of rice and corn in Davao del Sur since it provides an estimate to their possible yield. Thus, preparation for harvest season or planting season can be easily responded to.
- Merchants. Having a projection of the rise and fall of production rate can allow merchants to predict when the supply and possibly prices will fluctuate. This can allow them to have the opportunity to preemptively import or export more products based on the forecasted yield based on the model.
- The legislature. The study aims to provide a tool for projecting the production rate of corn and rice in Davao del Sur. This tool is helpful as a point of reference should lawmakers decide to implement new laws regarding production such as import and export.
- Future researchers. The results of the study can be grounds for further study. The study can be expanded to include the rice and corn production from other regions of the Philippines or the entire country itself. In addition, it can also be the foundation for future studies that involve other crops.

E. Scope and Limitations

1. The models is based on the data gathered from Davao del Sur only.
2. The dataset for this was obtained from the OpenSTAT website provided by the Philippine Statistics Authority (PSA).
3. The models that are used by the study are only SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM.

4. The models were developed using the programming languages Python and R. Existing Python packages for model development and visualization may be used to aid in development. In addition, some R libraries that may be helpful to the study can be used to make models. Thus, implementing it into the system also proves useful.

F. Assumptions

1. The dataset files are in csv format.
2. There are no missing values in the datasets.
3. The web application is deployed to the internet.
4. The users must have basic knowledge of computers to see the forecast.

II. Review of Related Literature

There have been some related research conducted in different countries around the world surrounding the forecasting of production. Varying methods are used, including ARIMA, Bayesian SARIMA, LSTM, and Exponential Smoothing, which are all related to the special problem. However these studies are not without limitations and constraints that differ with the current subject.

A. Related studies involving ARIMA

1. Empirical analysis for crop yield forecasting in India

A similar study by S. Dharmaraja et al. in 2020 [8] observed different methods of forecasting crop yield in India. For the purpose of the study, they used bajra (pearl millet) yield data in the Alwar district of Rajasthan from 1997 to 2016. The methods they used time series analysis models such as ARIMA and ARIMAX, and logistic regression such as simple and multiple regression. The study concludes that generally, time series analysis models are significantly better. In particular, the best performing model is the ARIMAX model.

This study aims to explore the crop yield not only using time series analysis models using ARIMA and ARIMAX, but also other models such as linear regression model using both simple regression and multiple regression. Since the topic also forecasts crop yield albeit a different plant, a lot of the methods used in this study would be applicable for the special problem. The study was successful in building a model that can make a projection for future crop yield with substantial confidence. This also opens the possibility of exploring further various other models that can be utilized and developed in forecasting such as the linear regression models. Should future researchers want to expand the scope of the special problem, then the methods introduced in this study can be the basis for alternative methods that they may use.

B. Related studies involving SARIMA

1. ARIMA model for forecasting of rice production in India by using SAS

One such study is by G. Ramakrishna, and R. V. Kumari in 2018 [7] aimed to develop a model based on ARIMA that can predict the amount of rice that India is able to produce in a yearly basis. This was done to give an estimate for their production rates in 2020. The study used a yearly rice production data between 1949 to 2016. It concluded that the best-fitting model is ARIMA(0,1,1), having the minimum AIC and BIC. The ACF and PACF of the residuals using the model had no pattern, thus provides a valid model for projection.

One key difference of the study from the special problem is its use of ARIMA instead of SARIMA. The former does not consider seasonality in its data. The dataset provided in the study only accounts for yearly rice production between 1949 and 2016. For this special problem, the dataset accounts for 1987 to 2022, but gathers data in a quarterly basis. Thus, there may be seasonal factors such as planting and harvest season and weather that are present in the special problem that are not used in the study. However, the study by G. Ramakrishna et al. used the appropriate model, since the nature of the dataset from their study does not use quarterly data.

Despite these differences, the study may still be beneficial to the special problem. This is because of their similarities in other aspects. Since both studies deal with time series analysis of rice involving a univariate dataset, practices and other methods from this study apart from the models can be used in developing the special problem, such as data preprocessing, analysis, among other things. This can provide a more grounded foundation for the structure of the special problem to have.

2. Forecasting on the crude palm oil production in Malaysia using SARIMA Model

Another study was made by S. A. M. Tayib et al. in 2021 [11] which tries to forecast the production of crude palm oil in Malaysia using the SARIMA model. The researchers obtained 5 years of monthly data of starting January 2014 up to September 2019 for a total of 69 entries. The final model that is most applicable is SARIMA(1, 0, 0)(0, 1, 1)₁₂. The actual values of the dataset are within 95% of the upper and lower confidence interval of the obtained model. Thus, the study concludes that the model is significant.

The study is an optimal basis for building the framework of the special problem because of its similarities. This study is arguably the closest study available that the special problem can follow. This is because the subject matter of the two are nearly identical. Both studies aim to forecast production yields of products that come from natural resources, albeit palm oil being a manufactured product and this process may influence the production rate. Both studies also utilize the SARIMA model instead of the ARIMA model. This is because the study uses monthly data that may be affected by seasonal changes such as reap and sow seasons as well as different weathers. Thus, the SARIMA model is more applicable for this topic, similar to the special problem.

C. Related studies involving ARMA

1. A guide to solar power forecasting using ARMA models

In 2019, B. Singh and D. Pozo [12] conducted involving the forecasting of solar power. The model they chose to use is the ARMA model. The study is not experimental, and the goal of the study is not to create the model itself. The study was instead made simply as a framework for developing a forecasting model using ARMA for future studies to reference to. The example that they have made can predict future solar power yield in an hourly manner.

This study provides an in-depth explanation to developing an ARMA model, which is one of the models to be used. Its step-by-step nature, especially in the methodology, will be very beneficial to the data analysis of the study once the results are already obtained. It provides a comprehensive view of the ARMA model, including the model itself, preprocessing of the data, validation of the stationarity of the time-series, parameter selection, and up to the prediction. While solar power and crop yield are two different types of products, the application is still applicable to since both deal with time series with cyclic datasets.

D. Related studies involving Holt-Winters

1. Forecasting of onion prices in Bangalore market: an application of time series models

Moving on to Exponential Smoothing methods, a study by Areef et al. in 2020 aimed to predict the future prices of onions in Bangalore. The study employs multiple different methods for doing so. These methods include ANN and SARIMA, as well as Exponential Smoothing methods such as Single ES, Double ES, and Holt-Winters ES. The dataset used was obtained from the NHRDF (National Horticulture Research and Development Foundation) in India which contains monthly prices from January 2003 to June 2018. The training set included the range Jan 2003 - Dec 2017, while the test set included from Jan 2018 - June 2018.

The top-performing models were chosen for each method first before comparing with the other models. For SARIMA, SARIMA(1, 1, 0)(1, 1, 1) was found to be the most effective having the highest R-square and lowest RMSE, MAPE, and MAE. For ANN, ANN₂ was found to be the most effective using the same metrics. For Exponential Smoothing, unfortunately, the Holt-Winters model was found to be the worst-performing, with the Single Exponential Smoothing as the most successful model. When comparing the three top performers among each other, ANN was the best-performing by having the least RMSE, MAPE, and MAE and most R-square value.

2. Modeling and forecasting number of confirmed and death caused COVID-19 in IRAN: A comparison of time series forecasting methods

An extensive study by N. Talkhi et al. [13] in 2021 uses many models in creating a forecast for COVID-19 confirmed cases and deaths. These methods include ARIMA, Holt-Winters, Hybrid, NNETAR, BSTS, TBATS, Prophet, MLP, ELM network. Among these methods, the most important to the special problem us are the Holt-Winters and ARIMA models. The dataset was taken from the Worldometers website. This included daily data from February 20 to August 15, 2020.

Between all the models they used, the best-performing for forecasting confirmed cases was the MLP network model, which had the lowest overall error. On the other hand, the Holt-Winters model was the best performing for forecasting the deaths. Having low errors in forecasting, the study concludes that both the MLP and Holt-Winters models are significantly effective in forecasting confirmed cases and death of COVID-19 in Iran.

3. Long-Term Forecasting of Electrical Loads in Kuwait Using Prophet and Holt–Winters Models

Another study which applied the Holt-Winters model is by A. Almazrouee in 2020 [14], which aims to forecast the peak loads of electricity consumption in Kuwait. The study uses two models: the Holt-Winters model and the Prophet model. The dataset was obtained from the Ministry of Electricity and Water of Kuwait which contains records from January 2010 to May 2020.

Both models performed well having low error metrics such as RMSE, MAE, MAPE, and high R^2 . Between the two models, Prophet performed slightly better, with an R^2 of 0.9942 compared to Holt-Winters' 0.9694. However, the difference between the two may be arguably negligible. The study concludes that both models are effective methods for forecasting peak electrical loads of Kuwait.

E. Related studies involving LSTM

1. Forecasting of rice cultivation in India - a comparative analysis with ARIMA and LSTM-NN models

For LSTM, there is a study which also uses rice production as its main subject matter. This is the study of K. K. Paidipati and Arjun Banik in 2019 [9]. For this study, the researchers decided to use ARIMA and LSTM-NN. The main objectives of this research paper are (1) to create a model; and (2) to compare the effectiveness of LSTM relative to ARIMA, since the former is a newer method. The dataset they used come from the Department of Agricultural and Cooperation in India, which contains univariate data 1950 from 2018. The elements they used are the Area Under Cultivation, Agricultural Production, and Agricultural Yielding. They used 1950-2005 as their training set and 2006-2018 as their testing/validation set.

The review will focus on the Agricultural Production since this is the variable most closely related to the special problem. The ARIMA model used for this is ARIMA(0,1,1). This model is proven to be effective in creating a forecast as it has an RMSE of 5.9814. However, the LSTM model outperforms the ARIMA model by having an RMSE of 3.447. Thus, the study concludes that LSTM is a viable alternative to traditional time series analysis methods such as ARIMA and can even have better results given the right amount of data. This study is a good starting point for the special problem in using LSTM as it uses the same subjects (rice production), as well as compare it with the traditional time series analysis such as ARIMA.

2. A Comparison of ARIMA and LSTM in Forecasting Time Series

Another study exists which aims to compare the performance of ARIMA and LSTM models. Siami-Namini et al. conducted a study which applies both ARIMA and LSTM models to the prices of certain commodities. The datasets were obtained from Yahoo! Finance, the Federal Reserve Bank of St. Louis, and the

International Monetary Fund (IMF) Website. [10] While the study also creates models based on the two methods, it focuses more on the improvement to the forecast when using LSTM instead of ARIMA.

As the study expects, LSTM performed considerably better than ARIMA. The average RMSE of the models when given the stock market data for LSTM is 64.213, while for ARIMA, 511.481. On the other hand, given economic-related data, LSTM has an RMSE of 0.936 while ARIMA has an RMSE of 5.999. The observed data shows an improvement of up to 87% when using LSTM instead of ARIMA. Thus, the study concludes the effectiveness deep-learning-based algorithms in dealing with time series data contrast to using only traditional Box-Jenkins methods. This gives the special problem grounds to explore the capabilities of LSTM when using different data.

3. LSTM Neural Network Based Forecasting Model for Wheat Production in Pakistan

In 2019, S. A. Haider et al. [2] explored the ability of LSTM. The goal of the study was to create a forecasting model using LSTM that can make future predictions to the wheat production in Pakistan. To help assessing the effectiveness of LSTM, ARIMA and RNN models were also created as comparison. The datasets used were obtained from the Federal Bureau of Statistics and the Economic Survey of Pakistan in 2017. This dataset includes wheat production history from 1902 up to 2018. Based on this, the records from 1902-2008 were used as training set, while 2009-2018 were used as the test set.

The study compares each of the three methods by testing them for both the raw data and the preprocessed data. From this, the model with the least RMSE, MAE, and R-value is determined to be the best-performing among the three. The results of the study show that the most effective of the three was the LSTM with preprocessed data, followed by ARIMA, then RNN. LSTM had the least RMSE and MAE with 792 and 729 respectively. Thus, the study concludes the

effectiveness of LSTM as a competent new method of analyzing time series data.

F. Conclusion

In conclusion, recent studies have proven the effectiveness of using time series analysis in forecasting production rates. The validity of SARIMA, Bayesian SARIMA, LSTM, and Winters have been used in current papers with meaningful success. However, there are some constraints in the local context that limit the results of these studies. In particular, there have been no recent studies in the Philippines that analyze local crop production using these models on existing local datasets. Furthermore, some of the studies presented used yearly obtained data with ARIMA. This can be improved by introducing seasonality, since the rice and corn production dataset present in Davao del Sur include quarterly data. Thus, SARIMA could be better implemented. This special problem aims to resolve these limitations as well as promote the development of data analytics in the Philippines particularly in the field of agriculture.

III. Theoretical Framework

A. Time Series Analysis

Current technologies allowed for the development of time series analysis. Through this, not only trends based on existing data are observed, but also predictions for future years that have not taken place yet.

Time Series Analysis is a modern way of analyzing trends in given a particular dataset, usually univariate. Unlike older forms of statistical treatment, Time Series Analysis involves inspecting currently available data to attempt to make a prediction about the future. This process is called forecasting. Forecasting is the creation of a model using finite past observations to make infinite future predictions. This method is most used in economics through market analysis. Even so, Time Series Analysis is a broad topic that can be applied to other fields of statistics. [15]

There is no concrete sample size requirement for all time series analysis methods. However, J. Hanke and D. Wichern [16] suggests a minimum of 50 for general time series methods. For Box-Jenkins models, which includes SARIMA and ARMA, the suggested minimum number of observations is 50. [17]. For Exponential Smoothing, no study was found. Thus, it is assumed that the 50 suggested by J. Hanke and D. Wichern will be sufficient. For LSTM, according to A. Alwosheel, the minimum sample size is dependent on the parameters that will be used. [18]

B. Autoregressive Moving Average (ARMA) Model

One of the methods of forecasting using Time Series Analysis is using Autoregressive Moving Average model (ARMA). ARMA models consists of two preexisting models [19]:

1. Autoregressive (AR) - The AR model obtains the present or future values using the regression of previous data points. The formula for the AR model

is as follows. The parameter p is decided by the researcher.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t \quad (1)$$

2. Moving Average (MA) - The MA model takes attempts to make a model using the error of the previous data points instead. The formula for the MA model is as follows. The parameter q is decided by the researcher.

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q} \quad (2)$$

The ARMA model incorporates both factors. Combining the two, an effective model can be made that takes the minimum number of parameters without sacrificing accuracy. The formula is simply a combination of the two models:

$$y_t = c + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_q \epsilon_{t-q} + \epsilon_t \quad (3)$$

ARMA has two parameters p and q which is decided by the researcher, yielding ARMA(p, q), which are as follows:

1. p : the number of items to consider for the autoregressive part
2. q : the number of items to consider for the moving average part

C. Autoregressive Integrated Moving Average (ARIMA) Model

Another existing method is by using the Autoregressive Integrated Moving Average model (ARIMA). Stationarity in a dataset represents no change in the mean or variance over time. Thus, cyclic time series (i.e. those with seasonal trends) can possibly still be stationary. Unlike ARMA, ARIMA attempts to remove non-stationarity in a time series. Thus, ARIMA is capable of modeling using both

stationary and non-stationary datasets. ARIMA has three parameters p , d , and q . An ARMA(p, q) model is equivalent to an ARIMA($p, 0, q$) model.

1. p : the number of items to consider for the autoregressive part
2. d : degree of differencing
3. q : the number of items to consider for the moving average part

D. Seasonal ARIMA (SARIMA) Model

A variation of ARIMA called Seasonal ARIMA (SARIMA) can be used. This model considers cyclic behavior. Thus, datasets with seasonality such as quarterly production of crops are more applicable to this model. [20] SARIMA is in the form SARIMA(p, d, q)(P, D, Q) $_s$, where:

1. p : the number of items to consider for the autoregressive part
2. d : degree of differencing
3. q : the number of items to consider for the moving average part
4. P : seasonal the number of items to consider for the autoregressive part
5. D : seasonal degree of differencing
6. Q : seasonal the number of items to consider for the moving average part
7. s : length of one season

E. Bayesian SARIMA Model

Bayesian SARIMA is a modification to the SARIMA model. It has similar parameters to the SARIMA model and only differs in parameter estimation. In Bayesian SARIMA, the parameters are assumed to be a random variable with its own distribution instead of being a constant. Thus, the obtained parameter is simply an estimate.

F. Holt-Winters Exponential Smoothing

The Holt-Winters Exponential Smoothing model is an extension to the Simple Exponential Smoothing (SES) model. The SES can take a univariate dataset and make a singular prediction for for upcoming values based on the past records [21]. While the SES can only create predictions based on past records, the Holt-Winters model also takes into account other factors that may be present in the time series such as trend and seasonality. The formula for getting the current forecast is as follows [22]:

$$F_{t+k} = L_t + kT_t + S_{t+k-M} \quad (4)$$

with components:

$$L_t = \alpha \frac{Y_t}{S_{t-M}} + (1 - \alpha)(L_{t-1} + T_{t-1}) \quad (5)$$

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \quad (6)$$

$$S_t = \gamma \frac{Y_t}{L_t} + (1 - \gamma)S_{t-M} \quad (7)$$

where:

1. F_{t+k} is the forecast.
2. L_t is the level.
3. T_t is the trend.
4. S_t is the seasonality, with M seasons.
5. k is the number of forecasts.

G. Long Short-Term Memory Model

Long Short-Term Memory (LSTM) Model is a modified form of the Recurrent Neural Network (RNN). LSTM is capable of making predictions given an input similar to any neural network with the exception of having the ability to retain

biases from previous iterations. Thus, it is found to be effective in time series analysis such as forecasting. It addresses one of the problems faced with RNN by separating the retained values into long-term and short-term memories to prevent gradient vanishing. [23]

An LSTM model is comprised of multiple LSTM-cells units which can be grouped into three parts: input gate, forget gate, and output gate. For every recurrence in the LSTM cell:

1. The input gate determines the inputs from both new data and the retained memory, as well as the processing necessary to get new values for the forecast, short-term, and long-term memories.
2. The forget gate determines how much of the long-term and short-term memory is retained.
3. The output gate returns the output that will be used for the next iteration of the LSTM cell.

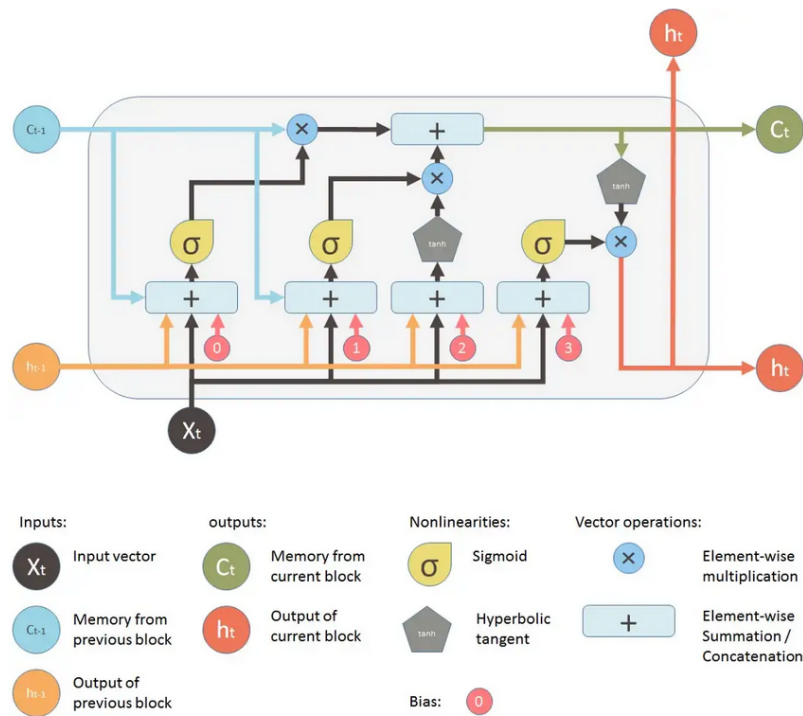


Figure 1: A diagram representing an LSTM Cell [1]

H. Accuracy Metrics

Four accuracy metrics will be used: Mean Absolute Deviation (MAD), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE).

The Mean Squared Error (MSE) is the average of the squares of the errors in the time series. It is one of the most commonly used measures of accuracy. This method takes into account the error for every point in the time series and is a measure of how close the predicted values are to the actual values.

$$MSE = \frac{\sum_{t=1}^n (A_t - F_t)^2}{n - 1} \quad (8)$$

The Root Mean Squared Error (RMSE) is simply the root of the MSE. The advantage of the RMSE is that the error is scaled to the units used in the dataset. Thus, it is easier to interpret. The RMSE can be interpreted as the average deviation between the actual values and the predicted values.

$$RMSE = \sqrt{MSE} \quad (9)$$

The Mean Absolute Deviation (MAD) is the average of the absolute values of each error. Unlike the MSE, each error is not squared. This is because the factor being minimized using the MSE is the mean alone, while the MAD uses the median. Thus, the MAD is better suited for forecasts that use predict using the median. This makes it less strict towards large errors compared to the MSE and RMSE.

$$MAD = \frac{\sum_{t=1}^n |A_t - F_t|}{n} \quad (10)$$

The Mean Absolute Percentage Error (MAPE) is the average of the percentage error of each prediction. MAPE is the same as MAD in the sense that it takes the absolute error of each entry, but differs with MAD by returning a percentage. Since the error returns a percentage, it is not scaled nor limited to units of the

data. This allows it to be comparable with other models that are scaled with different units.

$$MAPE = \frac{\sum_{t=1}^n |A_t - F_t|}{A_t} \times 100 \quad (11)$$

I. Existing Libraries

In applying these models, existing libraries can be used. For example, a paper by Fulton in 2022 [24] discusses the Python library statsmodels for forecasting time series using Bayesian estimation. The study is not experimental. It simply describes how to use the statsmodels library and the different models it can create based on the user's input. It also includes examples of these models in action using their dataset. For the examples, they use Bayesian estimation for approximating the parameters of the time series models. At the end of the paper, it concludes that Bayesian estimation is very possible using statsmodels. These libraries can be helpful for developing the topic, since one of the models to be used in the special problem is the Bayesian SARIMA. Using existing libraries will reduce the time it takes researchers manually creating the algorithms from scratch. For ARIMA models, libraries are also present for modelling, as demonstrated by A. S. Ahmar and A. Saleh et. Al in 2018 [25] in R for forecasting CPI data.

IV. Design and Implementation

A. Use Cases

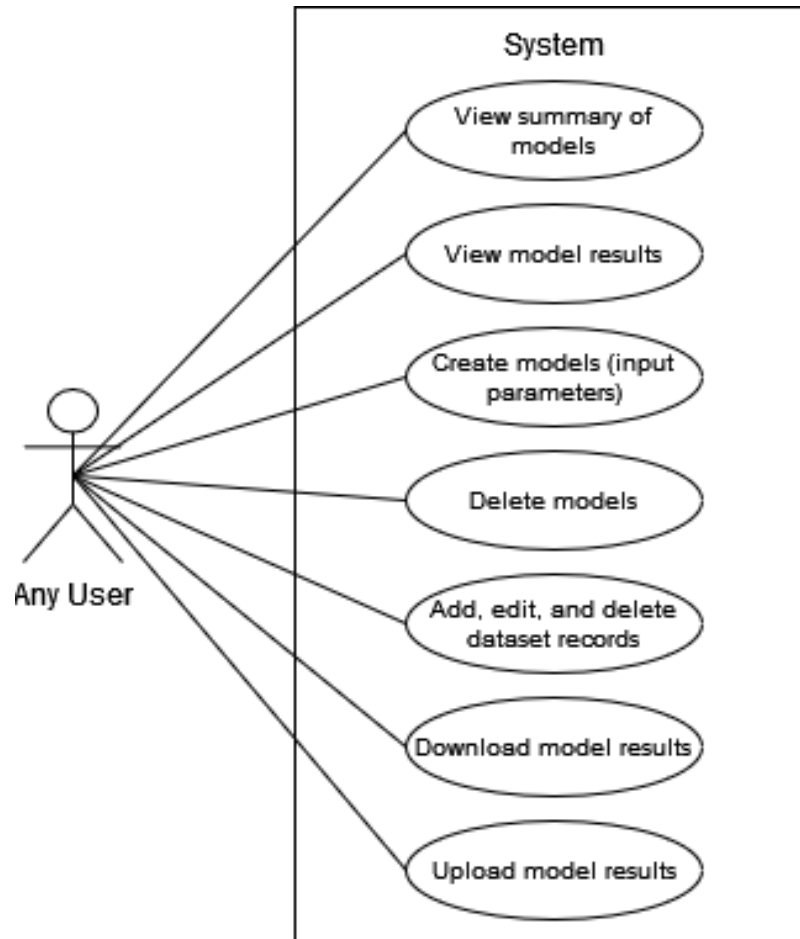


Figure 2: Use Case Diagram

The figure displays the use case diagram for the system. There is only one type of user, who has access to all of the functions available. The user is able to see the summary of all models existing in the session. The details of each model is also listed in the system. By default, the best models found by the researchers are displayed in the system.

The user can also submit their own models. Each model type can be added, with the user specifying the parameters they intend to use. Forecasts of each model can also be viewed by the user. The user also has the option to save the results into a JSON file, which can be reuploaded into the system.

The dataset in the system can also be modified should there be changes or new records in the data. The user can add, edit, and delete records for both rice and corn datasets.

B. The Dataset

The dataset was obtained from the OpenSTAT website by the Philippine Statistics Authority (PSA). OpenStat is a publicly-available database created by the PSA to provide various datasets that may be helpful to researchers, such as population, economy, agriculture, among many others. The website provides not only aggregated records for the entire Philippines, but also its components. This was used to get the dataset for the study.

From this database, two time series were obtained for Davao del Sur: one for rice and one for corn. Each set contains univariate quarterly records from the first quarter of 1987 until the last quarter of 2022, for a total of 144 records for each. Each dataset is in the form of a CSV file. The CSV file contains two columns: the starting date of each quarter, and the volume of production in that quarter.

C. Experimental Setup

The models that were used are SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM. The dataset was split into training and testing groups with roughly 90:10 ratio, resulting in 1987-2020 as the training set and 2020-2022 as the testing set. Each model uses the same metrics to compare with each other, such as the Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), and Mean Absolute Deviation (MAD). The model with the least values of each of the metrics is considered the best-performing model among the others.

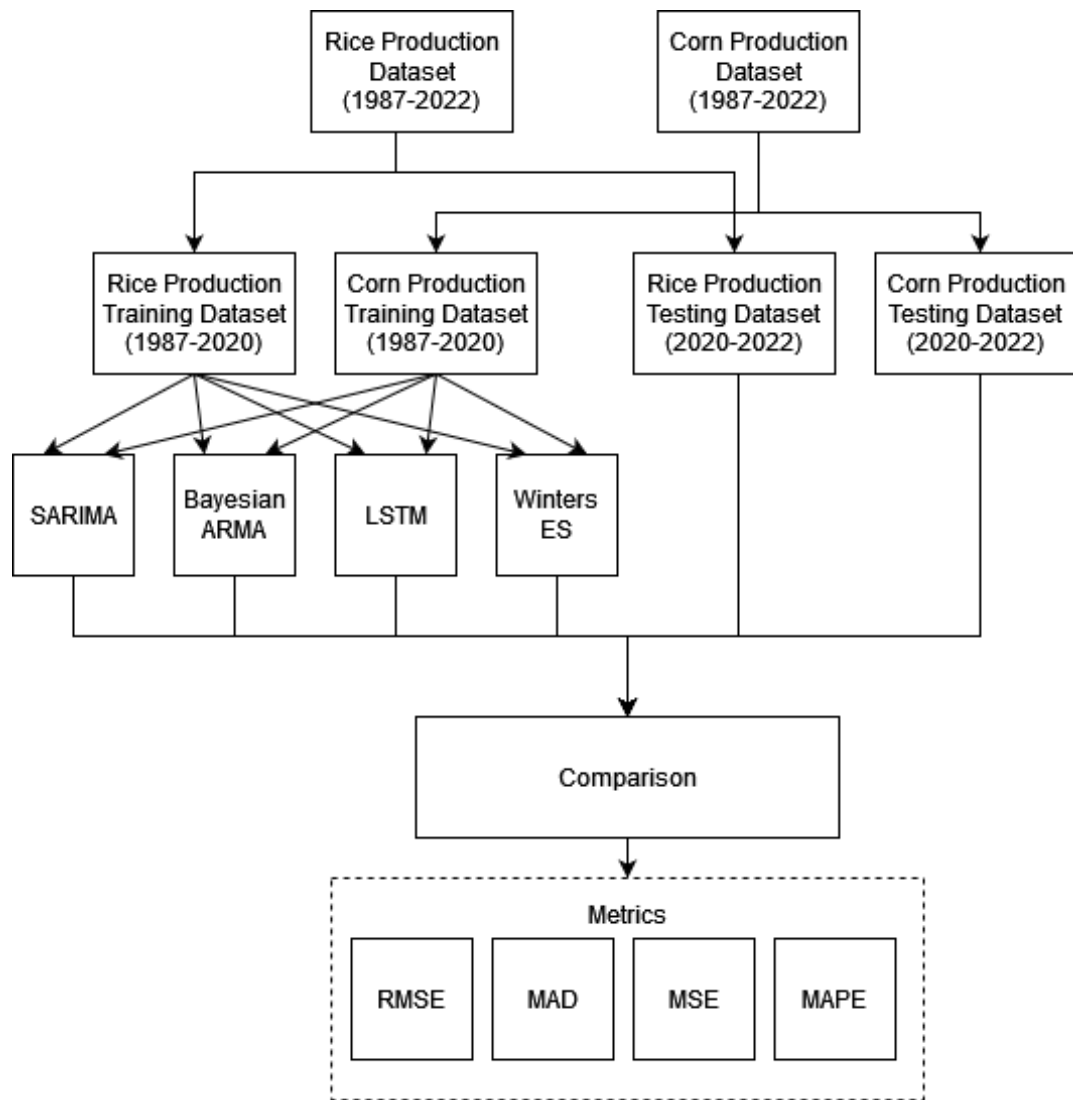


Figure 3: Diagram of the experimental setup based on Haider[2]

1. SARIMA

SARIMA is an adaptation to ARIMA models when the data is found to have seasonality. In this study, since the subject in question is crop production, factors such as growing and harvest seasons have a distinct effect in the time series. Thus, SARIMA is preferred over ARIMA. SARIMA takes the form of $SARIMA(p, d, q)(P, D, Q)_s$. To estimate both seasonal and non-seasonal parameters that will be used in the model, ACF and PACF plots were used. This ensures that the chosen model has the most significance among the others. After obtaining the model, the researchers can proceed with model checking, where the model is applied to test group. Lastly, forecasting can now be done, which is compared to the other models used in the study.

To create the SARIMA model, the `Arima` function from the `stats` R library was used. Then, the R codes were translated to fit into Python syntax and imported using the `rpy2` package. The parameters that were used are the order and seasonal order.

2. Bayesian SARIMA

Bayesian SARIMA is the use of Bayesian Statistics in estimating the parameters of the SARIMA model. The initial parameters were based on the best model found using SARIMA. Then, the parameters are estimated using the training set. After the parameters are estimated, the test set will be input into the obtained model. The fitted values will be compared with the actual values to test the effectiveness of the model.

To simplify this process, the system uses `bayesforecast`. `bayesforecast` is an R library that implements all of the necessary methods for Bayesian Forecasting by using the programming language Stan. Stan is a probabilistic language that can be implemented in R using the package `rstan`. The function `stan.sarima` was used in creating the model. Similar to SARIMA, implementing R codes into python were done using the `rpy2` package.

3. Long Short-Term Memory

The LSTM model is an improvement to the RNN model that solves the gradient vanishing problem brought about the RNN model.

In the study, existing libraries may be used in model development. For LSTM, the Keras Python deep learning API developed by F. Chollet is applicable. The library contains `keras.layers.LSTM` which were used to simplify the process. The parameters that were used are the number of units, number of epochs, and the window size.

4. Holt-Winters Exponential Smoothing

The Holt-Winters model is an extended form of the Simple Exponential model, which adds seasonality and trend to the latter to accommodate seasonal univariate datasets. The formula of Holt-Winters is as follows:

$$F_{t+k} = L_t + kT_t + S_{t+k-M} \quad (12)$$

with components:

$$L_t = \alpha \frac{Y_t}{S_{t-M}} + (1 - \alpha)(L_{t-1} + T_{t-1}) \quad (13)$$

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \quad (14)$$

$$S_t = \gamma \frac{Y_t}{L_t} + (1 - \gamma)S_{t-M} \quad (15)$$

For the development of the Winters model, existing libraries were be used. One such library is `statsmodels`, which is a free package that can be used to create statistical models. In this library, the `statsmodels.tsa.holtwinters` class can be used.

Given the nature of the Holt-Winters package, the parameters changed were the type of Holt-Winters to be used (additive or multiplicative), and damping.

D. System Architecture

The web application is implemented using the Django Framework from Python. The SARIMA and Bayesian SARIMA models use `stats` and `bayesforecast` respectively from R. Thus, R is also needed in the system, with `rpy2` being the interface. Lastly, the Holt-Winters model uses `statsmodels`, while the LSTM model uses `keras`. Other helper packages were also used in handling the data such as `pandas`, `numpy`, and `scikit-learn`, and visualization using `matplotlib`

1. Python 3.11.3
 - (a) Django 4.1.7
 - (b) statsmodels 0.13.5
 - (c) keras 2.12.0
 - (d) numpy 1.23.5
 - (e) scikit-learn 1.2.2
 - (f) matplotlib 3.7.0
 - (g) rpy2 3.5.1
2. R 4.3.0
 - (a) stats 4.3.0
 - (b) forecast 8.21
 - (c) bayesforecast 1.0.1
 - (d) rstan 2.21.8

E. Technical Architecture

The web application is deployed using Docker. It was built using the Python base image `python:3.11.3-bullseye`. The image runs on Debian 11.7, with Python 3.11.3 installed. After this, `r-cran-rstan 2.21.8-1` was installed in the system

to accommodate the R and `rstan` dependencies. The requirements enumerated are as follows:

1. 2 GHz processor
2. 6 GB disk space
3. 4 GB of RAM
4. Debian-bullseye 11.7 OS

Since the system is a web application, most of the computing is done by the server. Thus, minimal resources are needed when used by the user.

1. 2 GHz processor
2. 1 GB disk space
3. 2 GB of RAM
4. Any OS
5. Web Browser

V. Results

A. Landing Page

Figure 4 shows the index page of the web application. This serves as the first page that the user sees upon entering the website. The landing page shows the basic information of the system such as the title and description.



Figure 4: Landing Page

B. Model Page

Upon entering the website, the user can view a summary of all current models in the system as in Figure 5. By default, the displayed models are the best models found by the researchers.

On the left side is a display of all existing models for the dataset superimposed into a single figure. It displays the test set, colored blue, with the highest line weight, and the other models as displayed in the legend. It also shows forecast of up to 2024 by default, but this can be extended up using the form below the graph.

There are also four buttons that the user can interact with:

1. The reset button resets everything about the current dataset to its initial state. This includes deleting user-made models and reloading the re-

searchers' models and the dataset in question.

2. The upload button allows the user to submit a file that will be the basis of creating a model. The file can be obtained by downloading a model from this website.
3. The summary button displays a table of the accuracy metrics of each model as seen in Figure 6. This includes the BIC, MSE, RMSE, MAD, and MAPE.
4. the edit dataset button directs the user to the edit dataset page, where the user can view and modify the records in the stored dataset.

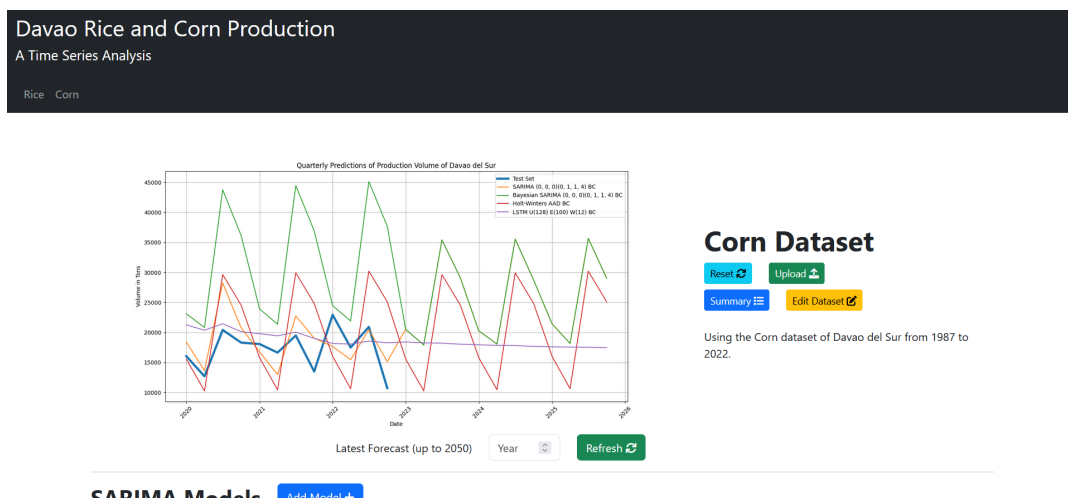


Figure 5: Graphical summary of all models

| Forecasts | | | | | |
|--|--------------------|----------------------|---------------------|---------------------|--------------------|
| Model | BIC | MSE | RMSE | MAD | MAPE |
| SARIMA (0, 0, 0)(0, 1, 1, 4) BC | 1,655.987228847331 | 15,131,606.827907069 | 3,889.9366097543375 | 3,288.8111279278014 | 19.865551576128333 |
| Bayesian SARIMA (0, 0, 0)(0, 1, 1, 4) BC | -- | 295,092,944.08514225 | 17,178.2695311589 | 14,369.567437644024 | 90.05215490804986 |
| Holt-Winters AAD BC | -- | 72,585,647.943928 | 8,519.721118905713 | 7,170.847033821791 | 44.68208340947892 |
| LSTM U(128) E(100) W(12) BC | -- | 19,771,907.86036003 | 4,446.561352366572 | 3,465.0950074429297 | 23.62855962702547 |

Figure 6: Tabulized summary of all models

Below the summary section are the individual models saved in the system. The section displays a plot of the entire dataset, the predicted values for the test set,

and the forecasted values. Similar to the summary, the range of prediction can be chosen in this section.

Other relevant information are also displayed next to the model. This includes the parameters used and the accuracy metrics. Each model type displays different information since each has different parameters. For example in Figure 7, the following are details of a SARIMA model.

There are also 3 buttons that the user can interact with:

1. The save button allows the user to download a JSON file that contains information about the model. This can be submitted using the upload function to restore a model.
2. The forecast button displays a table of the forecasted values and its respective errors as seen in Figures 8.
3. The delete button deletes the model.

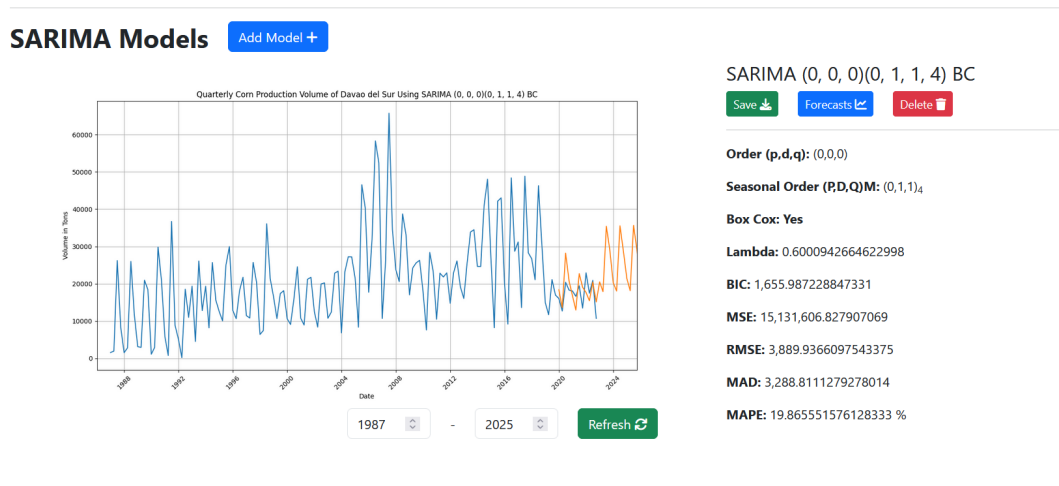


Figure 7: Summary of a SARIMA model

The user can also create their own model from scratch. This prompts a popup window in which the user can provide the parameters they wish to use, as seen in Figure 9.

| Period | Actual Value | Forecast | Error |
|---------|--------------|---------------------|----------------------|
| 2020 Q1 | 16,061.62 | 18,417.69109514514 | -2,356.0710951451383 |
| 2020 Q2 | 12,697.37 | 13,623.72641610743 | -926.35641610743 |
| 2020 Q3 | 20,419.05 | 28,229.738936604164 | -7,810.688936604165 |
| 2020 Q4 | 18,310.85 | 20,777.885835212346 | -2,467.0358352123476 |
| 2021 Q1 | 18,072.0 | 16,779.659948528177 | 1,292.3400514718232 |
| 2021 Q2 | 16,644.38 | 12,982.328880677722 | 3,662.051119322279 |
| 2021 Q3 | 19,496.0 | 22,736.77592020717 | -3,240.7759202071684 |
| 2021 Q4 | 13,500.0 | 19,063.817518039723 | -5,563.817518039723 |
| 2022 Q1 | 22,941.67 | 17,666.459051937232 | 5,275.210948062766 |
| 2022 Q2 | 17,491.0 | 15,468.051435136298 | 2,022.948564863702 |
| 2022 Q3 | 20,922.65 | 20,481.314355111524 | 441.33564488847696 |

Figure 8: Forecasted values of a SARIMA model

Add a SARIMA Model

Order:

P* This field is required.

D* This field is required.

Q* This field is required.

Seasonal Order:

Seasonal P* This field is required.

Seasonal D* This field is required.

Seasonal Q* This field is required.

Season M* This field is required.

Transformation:

Use Box Cox Transformation

Lambda (default: auto)

Close X Save changes




Figure 9: Add model popup

C. Dataset Page

The dataset page displays the entire dataset of the chosen crop. In this page, the user has multiple options:

1. The reload dataset button resets the dataset to its original state.
2. The test set year field sets the start year of the test set. By default, this is 2020.
3. The edit button shows a popup to edit the current record.
4. The add button shows a popup to add a new record. This record is automatically added to the end of the time series.
5. The delete button deletes the last record in the dataset.

Corn Dataset

Reload Dataset  Test Set Year: 2020  Save 












| Quarter | Volume | |
|---------|---------|--|
| 1987 Q1 | 1580.0 | Edit  |
| 1987 Q2 | 1960.0 | Edit  |
| 1987 Q3 | 26230.0 | Edit  |
| 1987 Q4 | 8440.0 | Edit  |
| 1988 Q1 | 1537.0 | Edit  |
| 1988 Q2 | 2823.0 | Edit  |
| 1988 Q3 | 26010.0 | Edit  |

Figure 10: Edit dataset page

| | | |
|---------|----------|--|
| 2022 Q1 | 22941.67 | Edit  |
| 2022 Q2 | 17491.0 | Edit  |
| 2022 Q3 | 20922.65 | Edit  |
| 2022 Q4 | 10738.84 | Edit  |



Add a record  Delete last record 

Figure 11: Edit dataset cont.

VI. Discussions

A. Dataset

The format of the datasets for rice and corn are identical. Each contains the volume of the quarterly production of the crops in Davao del Sur from 1987 to 2022. The data is univariate and has a total of 144 records. There were no missing values from the dataset.

The dataset was split into train and test sets. The train set contained data from the first quarter of 1987 until the last quarter of 2019. The test set contained data from the first quarter of 2020 until the last quarter of 2022. This results in a 90:10 test split.

One option for analysing time series datasets is using the Box-Cox Transformation. This allows the residuals of the data to be normalized, and can be a way to improve predictions. The Box-Cox transformation was applied to the datasets before each model was created.

B. Model Development

Four models were used in the study: SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM. Each model was given the train set to develop a model based on the different parameters it requires. The parameters for SARIMA were chosen using the BIC of the dataset. Whichever set of order and seasonal order has the lowest BIC was used. Similarly, Bayesian SARIMA used the same order obtained in SARIMA for comparison. For the Holt-Winters model, the parameters were whether to use additive or multiplicative for the trend and seasonal components and if damping is necessary. For LSTM, different number of units, number of epochs, and windows size were used. For both Holt-Winters and LSTM, the fit with the best accuracy metrics are determined to be the best fitting models, with the lowest being accepted.

For the rice dataset, the best-performing SARIMA model is $\text{SARIMA}(1, 0, 1)(0, 1, 1)_4$

| Rice | SARIMA | Bayesian SARIMA | Holt-Winters | LSTM |
|------|------------|-----------------|---------------|---------------|
| MSE | 9801282.49 | 26,966,930.43 | 25,602,666.24 | 35,237,159.42 |
| RMSE | 3130.7 | 5,192.97 | 5,059.91 | 5,936.09 |
| MAD | 2,557.38 | 4,026.39 | 4,199.35 | 4,513.46 |
| MAPE | 8.42% | 11.43% | 14.14% | 16.52% |

Table 1: Accuracy metrics for rice

with a BIC of 597.2791. This order and seasonal order were used for both SARIMA and Bayesian SARIMA models. Using Holt-Winters, the most accurate model was using multiplicative methods for both trend and seasonality, and no damping, with an RMSE of 4199.35. Lastly, the best LSTM model used 128 units, 100 epochs, and a window size of 12.

On the other hand, the corn dataset had different results. The best-performing SARIMA model is SARIMA(0,0,0)(0,1,1)₄ with a BIC of 1655.987. This order and seasonal order were used for both SARIMA and Bayesian SARIMA models. Using Holt-Winters, the most accurate model was using additive methods for both trend and seasonality, and and damping, with an RMSE of 8,519.72. Similar to the rice mode, the best LSTM model also used 128 units, 100 epochs, and a window size of 12.

1. Accuracy Metrics

The predictions for the rice and corn datasets can be seen in Tables 1 and 3 respectively. The predictions were compared to the actual values of the test set. The resulting metrics are displayed in Table 2 and 4 respectively.

In the rice dataset, the obtained models are slightly close to each other, with the MAPE ranging only from 8-17%. It is also visible on the superimposed graph. The best-performing model was SARIMA, with an MSE of 9801282.49, RMSE of 3130.7, MAD of 2557.38, and MAPE of 8.42%. The worst-performing was the LSTM, with an MSE of 32237159.42, RMSE of 5936.09, MAD of 4513.46, and MAPE of 16.52%.

On the other hand, the corn dataset is more varied. the lowest MAPE is 19.87%

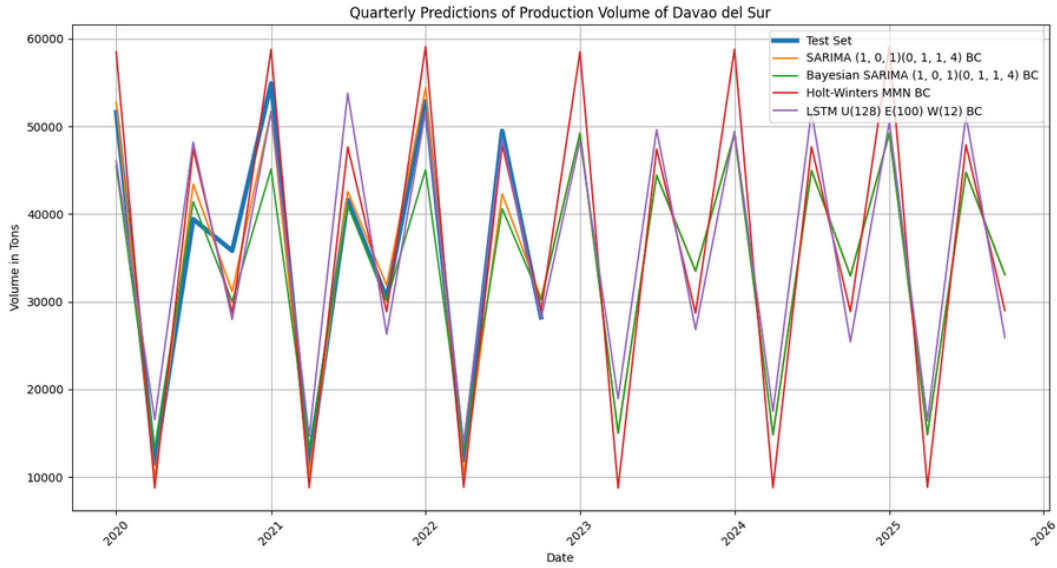


Figure 12: Actual and fitted values of rice for each model

from SARIMA and as high as 90.05% for Bayesian SARIMA. This is also visible on the superimposed plots. The SARIMA, Bayesian SARIMA, and Holt-Winters models show signs of seasonality, while the LSTM is almost smooth and flat.

Similar to the rice dataset, the best-performing model was still SARIMA. Its MSE was 15142729.35, RMSE of 3891.366, MAD of 3289.88, and MAPE of 19.87%. The worst-performing was the Bayesian SARIMA, with an MSE of 295092944.09, RMSE of 17178.27, MAD of 14369.57, and MAPE of 90.05%.

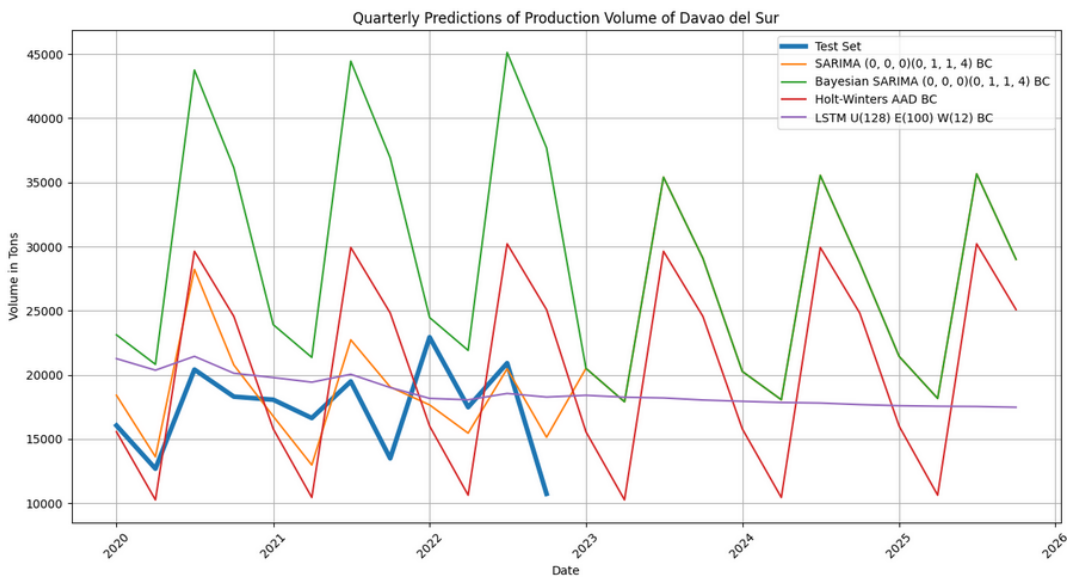


Figure 13: Actual and fitted values of corn for each model

| Corn | SARIMA | Bayesian SARIMA | Holt-Winters | LSTM |
|------|-------------|-----------------|---------------|---------------|
| MSE | 15142729.35 | 295,092,944.09 | 72,585,647.94 | 19,771,907.86 |
| RMSE | 3891.366 | 17,178.27 | 8,519.72 | 4,446.56 |
| MAD | 3,289.88 | 14,369.57 | 7,170.85 | 3,465.10 |
| MAPE | 19.87% | 90.05% | 44.68% | 23.63% |

Table 2: Accuracy metrics for corn

Based on the accuracy metrics, the MSE is difficult to interpret because of its squared nature. However, using RMSE simplifies this. For example, the RMSE for rice being 3891.366 for SARIMA means that the deviation between each predicted and actual value is about 3891.366 tons. On the other hand, the MAD is more lenient towards outliers. This explains the closeness of the MAD of each model. Lastly, the MAPE represents a percentage of how far each error is from the actual value, regardless of the unit of measure used. This allows the performance of the rice and corn models to be compared from each other. Based on the results, rice has lower MAPE for each model. Thus, it can be concluded that the models created for rice are more accurate.

2. Forecasts

For both rice and corn dataset, there is minimal trend in the forecasts. The predicted results tend to increase almost negligibly over the year. All models appear to have seasonality, except for the forecasts of the LSTM in the corn dataset, which appears to be flat.

C. Discussion

In general, the SARIMA model appears to be the top-performing for the datasets. This is different from the findings of other researchers, since most findings tend to have the LSTM perform the best. For example, a study by Paidpati and Banikk [9] which compared LSTM and ARIMA, another Box-Jenkins model. Based on their study, there is at least a 4% improvement in the MAPE, LSTM with 1.3686 compared to ARIMA with 5.6035. A different study is by Siami-Namini[10], which

| Period | SARIMA | Bayesian SARIMA | Holt-Winters | LSTM |
|---------|-----------|-----------------|--------------|-----------|
| 2023 Q1 | 49,226.44 | 49,226.44 | 58,512.45 | 48,373.89 |
| 2023 Q2 | 15,006.40 | 15,006.40 | 8,731.23 | 18,929.13 |
| 2023 Q3 | 44,434.33 | 44,434.33 | 47,421.70 | 49,629.95 |
| 2023 Q4 | 33,477.57 | 33,477.57 | 28,704.05 | 26,818.98 |
| 2024 Q1 | 49,181.03 | 49,181.03 | 58,809.16 | 49,420.63 |
| 2024 Q2 | 14,807.79 | 14,807.79 | 8,775.50 | 17,459.73 |
| 2024 Q3 | 44,971.99 | 44,971.99 | 47,662.16 | 51,655.61 |
| 2024 Q4 | 32,939.11 | 32,939.11 | 28,849.59 | 25,416.26 |
| 2025 Q1 | 49,206.69 | 49,206.69 | 59,107.37 | 50,475.08 |
| 2025 Q2 | 14,825.71 | 14,825.71 | 8,819.99 | 16,395.64 |
| 2025 Q3 | 44,768.98 | 44,768.98 | 47,903.84 | 51,229.29 |
| 2025 Q4 | 33,081.86 | 33,081.86 | 28,995.88 | 25,898.76 |

Table 3: Forecasts for rice until 2025

| Period | SARIMA | Bayesian SARIMA | Holt-Winters | LSTM |
|---------|-----------|-----------------|--------------|-----------|
| 2023 Q1 | 20,515.73 | 20,515.73 | 15,572.16 | 18,417.49 |
| 2023 Q2 | 17,910.09 | 17,910.09 | 10,266.65 | 18,266.06 |
| 2023 Q3 | 35,414.64 | 35,414.64 | 29,636.95 | 18,211.68 |
| 2023 Q4 | 29,091.63 | 29,091.63 | 24,561.93 | 18,047.16 |
| 2024 Q1 | 20,275.08 | 20,275.08 | 15,800.39 | 17,948.42 |
| 2024 Q2 | 18,069.92 | 18,069.92 | 10,454.30 | 17,857.39 |
| 2024 Q3 | 35,550.87 | 35,550.87 | 29,934.28 | 17,812.61 |
| 2024 Q4 | 28,727.83 | 28,727.83 | 24,832.49 | 17,693.98 |
| 2025 Q1 | 21,451.17 | 21,451.17 | 16,020.39 | 17,604.31 |
| 2025 Q2 | 18,172.11 | 18,172.11 | 10,635.46 | 17,560.67 |
| 2025 Q3 | 35,671.02 | 35,671.02 | 30,220.32 | 17,543.03 |
| 2025 Q4 | 29,007.41 | 29,007.41 | 25,092.89 | 17,485.19 |

Table 4: Forecasts for corn until 2025

shows an increase in the accuracy by up to 85%. For the Holt-Winters model, it performed decently well albeit not being the most accurate. This is backed by other related studies such as by Almazrouee[14] and Areef[26], where the Holt-Winters is not the best fit, but has low error metrics.

Apart from obtaining best-fitting models, part of the objectives of the study is the creation of a forecast system that can be accessed by all. The study meets this objective since the system is a web application open to anyone with internet access. The system serves as both a forecasting system for stakeholders that may be either related to statistics or crop production, but may not be both. People with a statistics background such as future researchers or other statisticians may use the system to develop their own models using the web application to create their own results. On the other hand, other stakeholders such as producers, the legislature, and the merchants can stick to the obtained models provided by the researcher. Thus, they will be able to see forecasts for the crops without any prior background in time series analysis.

VII. Conclusions

The web application developed in this study aims to conduct a time series analysis on the crop production of Davao del Sur. This was achieved by applying SARIMA, Bayesian SARIMA, Holt-Winters, and LSTM methods using the rice and corn productions of the region. In addition to finding the best models, the web application also has the tools in order for the user to develop their own using their own inputs.

Overall, the study achieves its utmost goal in the big picture– to increase awareness in applying statistical methods in applicable fields in the Philippines. Whether or not the best models were obtained, it proves that statistics can be applied outside of the academe. The system also encourages people interested in the topic both by making predictions and giving the users a chance to explore using the functions of the web application. With this much capability, it is essential that these methods are applied in the development of the country.

VIII. Recommendations

There are still possible improvements that can be done in the system. For example, the analysis may have some improvement if more preprocessing could have been done.

In addition, using a different dataset can be a way of expanding the research study. This can be in the form of using the rice and corn production of other regions in the Philippines or even the Philippines as a whole. Another way of expanding could be the use of different crops aside from rice and corn that are available from different sources that may be valuable in the Philippines.

Some improvements may be to use other algorithms, as many new forms of machine learning are developing over time. Due to time and resource constraints, the study was limited to only four. However, other possible methods that could be used are Prophet, ELM, and TBATS, as used by Talkhi [13].

IX. Bibliography

- [1] S. yan, “Understanding lstm and its diagrams,”
- [2] S. Haider, S. Naqvi, T. Akram, G. Umar, A. Shahzad, M. Sial, S. Khaliq, and M. Kamran, “Lstm neural network based forecasting model for wheat production in pakistan,” *Agronomy*, vol. 9, no. 2, 2019.
- [3] Department of Science and Technology - Food and Nutrition Research Institute, “Expanded national nutrition survey,” 2018.
- [4] Food and Agriculture Organization of the United Nations, “Crops and livestock products,” 2022.
- [5] Bureau of Plant Industry - National Plant Quarantine Services Division - Central, “2019-2022 rice arrival per origin (as of november 10, 2022).,” 2022.
- [6] F. A. S. U. D. of Agriculture, “Philippines: Grain and feed annual,” 2022.
- [7] G. Ramakrishna and R. V. Kumari, “Arima model for forecasting of rice production in india by using sas,” *International Journal of Applied Mathematics and Statistical Sciences (IJAMSS)*, vol. 6, no. 4, 2018.
- [8] S. Dharmaraja, V. Jain, P. Anjoy, and H. Chandra, “Empirical analysis for crop yield forecasting in india,” *Agricultural Research*, vol. 9, no. 1, 2020.
- [9] K. K. Paidipati and A. Banik, “Forecasting of rice cultivation in india—a comparative analysis with arima and lstm-nn models,” *EAI Endorsed Transactions on Scalable Information Systems*, vol. 20, no. 24, 2019.
- [10] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, “A comparison of arima and lstm in forecasting time series,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1394–1401, 2018.

- [11] S. A. M. Tayib, S. R. M. Nor, and S. M. Norrulashikin, “Forecasting on the crude palm oil production in malaysia using sarima model,” *Journal of Physics*, vol. 1988, 2021.
- [12] B. Singh and D. Pozo, “A guide to solar power forecasting using arma models,” *IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, 2019.
- [13] N. Talkhi, N. A. Fatemi, Z. Ataei, and M. J. Nooghabib, “Modeling and forecasting number of confirmed and death caused covid-19 in iran: A comparison of time series forecasting methods,” *Biomed Signal Process Control*, vol. 66, 2021.
- [14] A. Almazrouee, A. Almeshal, A. Almutairi, M. Alenezi, and S. Alhajerj, “Long-term forecasting of electrical loads in kuwait using prophet and holt-winters models,” *Applied Sciences*, vol. 10, no. 16, 2020.
- [15] C. Shetty, “Time series models,” *Towards Data Science*, 2022.
- [16] J. Hanke and D. Wichern, *Business Forecasting*. Pearson, 2008.
- [17] M. Malaya, “Forecasting in business research using the arima box-jenkins methodology,” *DLSU Business and Economics Review*, vol. 12, no. 1, 2001.
- [18] A. Alwosheel, “Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis,” *Journal of Choice Modelling*, vol. 28, pp. 167–182, 2018.
- [19] Z. Zhang and J. Moore, *Autoregressive Moving Average Models*. 2015.
- [20] R. Nau, “Introduction to arima: nonseasonal models,” *Statistical forecasting: notes on regression and time series analysis*, 2020.
- [21] E. Ostertagová and O. Ostertag, “The simple exponential smoothing model,” in *The 4th International Conference on modelling of mechanical and mecha-*

- tronic systems, Technical University of Košice, Slovak Republic, Proceedings of Conference*, pp. 380–384, 2011.
- [22] L. Ariton, “A thorough introduction to holt-winters forecasting,” 2021.
- [23] S. Hochreiter, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, 1997.
- [24] C. Fulton, “Bayesian estimation and forecasting of time series in statsmodels,” *Proceedings of the 21st Python in Science Conference*, 2022.
- [25] A. S. Ahmar, A. Saleh, and et al., “Implementation of the arima (p, d, q) method to forecasting cpi data using forecast package in r software,” *Journal of Physics: Conference Series*, vol. 1028, no. 1, 2018.
- [26] M. Areef, S. Rajeswari, N. Vani, and G. M. Naidu, “Forecasting of onion prices in bangalore market: an application of time series models,” *Indian Journal of Agricultural Economics*, vol. 75, no. 02, pp. 217–227, 2020.

X. Appendix

A. Source Code

1. Imports

```
import base64
from io import BytesIO
from datetime import datetime
import math
import csv
import json
import os

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pymc as pm
import seaborn as sns

import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

import rpy2
import rpy2.robjects as robjects
from rpy2.robjects.packages import importr, data

# reproducibility
import os
tf.keras.utils.set_random_seed(5)
os.environ['PYTHONHASHSEED']=str(5)
```

2. General Utils

```
def get_r_package(pkg_name):
    lib_dir1 = '/usr/local/lib/R/site-library'
    lib_dir2 = '/usr/lib/R/site-library'
    lib_dir3 = '/usr/lib/R/library'

    try:
        return importr(pkg_name, suppress_messages=False, lib_loc=lib_dir1)
    except:
        try:
            return importr(pkg_name, suppress_messages=False, lib_loc=lib_dir2)
        except:
            return importr(pkg_name, suppress_messages=False, lib_loc=lib_dir3)

def get_graph():
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
    image_png = buffer.getvalue()
    graph = base64.b64encode(image_png)
    graph = graph.decode('utf-8')
    buffer.close()
    plt.close()

    return graph

def plot_model(dataset_data, test_set_index, model):
    test_set_date = dataset_data.iloc[test_set_index]['Date']
    no_of_periods = (2050 - test_set_date.year + 1) * 4
    forecast_dates = pd.date_range(start=test_set_date, periods=no_of_periods, freq="QS")
    predictions = model['forecasts']

    plt.figure(figsize=[15, 7.5])
    plt.plot(dataset_data['Date'], dataset_data['Volume'])
    plt.plot(forecast_dates, predictions)
    plt.xlim(datetime(year=int(model['display_start']) - 1, month=1, day=1), datetime(
        year=int(model['display_end']), month=10, day=1))
    plot_title = 'Quarterly ' + model['dataset'] + ' Production Volume of Davao del Sur
        Using ' + model['model.name']
    plt.title(plot_title)
    plt.ylabel('Volume in Tons')
    plt.xlabel('Date')
    plt.xticks(rotation=45)
    plt.grid(True)

    return get_graph()

def get_merged_graphs(sarima_models, bayesian_models, winters_models, lstm_models, test_set,
    end_year):
    plt.figure(figsize=[15, 7.5])
    plot_title = 'Quarterly Predictions of Production Volume of Davao del Sur'
```

```

plt.title(plot_title)
plt.ylabel('Volume in Tons')
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.grid(True)

plt.plot(test_set['Date'], test_set['Volume'], linewidth=4, label="Test Set")
plt.legend()

date_start = test_set['Date'][test_set.index.start]
no_of_periods = (end_year - date_start.year + 1) * 4

if (len(sarima_models) + len(bayesian_models) + len(winters_models) + len(lstm_models)
    < 1):
    return get_graph()

for model in sarima_models:
    forecast_dates = pd.date_range(start=date_start, periods=no_of_periods, freq="
    QS")
    predictions = []
    for i in range(no_of_periods):
        predictions.append(model['forecasts'][i])

    plt.plot(forecast_dates, predictions, label="{0}".format(str(model['model_name
    '])))
    plt.legend()

for model in bayesian_models:
    forecast_dates = pd.date_range(start=date_start, periods=no_of_periods, freq="
    QS")
    predictions = []
    for i in range(no_of_periods):
        predictions.append(model['forecasts'][i])

    plt.plot(forecast_dates, predictions, label="{0}".format(str(model['model_name
    '])))
    plt.legend()

for model in winters_models:
    forecast_dates = pd.date_range(start=date_start, periods=no_of_periods, freq="
    QS")
    predictions = []
    for i in range(no_of_periods):
        predictions.append(model['forecasts'][i])

    plt.plot(forecast_dates, predictions, label="{0}".format(str(model['model_name
    '])))
    plt.legend()

for model in lstm_models:
    forecast_dates = pd.date_range(start=date_start, periods=no_of_periods, freq="
    QS")
    predictions = []
    for i in range(no_of_periods):
        predictions.append(model['forecasts'][i])

    plt.plot(forecast_dates, predictions, label="{0}".format(str(model['model_name
    '])))
    plt.legend()

return get_graph()

def reload_dataset(request, dataset):
    dataset_dates = "{0}_dataset_dates".format(dataset.lower())
    dataset_name = "{0}_dataset_data".format(dataset.lower())
    dataset_data = pd.DataFrame()

    request.session[dataset_dates] = []
    request.session[dataset_name] = []

    clear_all_models(request, dataset.lower())

    filename = "static/{0} data.csv".format(str.lower(dataset))
    with open(filename) as file:
        reader = csv.reader(file)
        readerlist = []
        next(reader)

        for row in reader:
            readerlist.append(row)

    dataset_data = pd.DataFrame(readerlist, columns=['Date', 'Volume'])
    dataset_data['Volume'] = pd.to_numeric(dataset_data['Volume'])
    dataset_data['Date'] = pd.to_datetime(dataset_data['Date'])

    request.session[dataset_dates] = dataset_data['Date'].astype(str).tolist()
    request.session[dataset_name] = dataset_data['Volume'].tolist()
    request.session.modified = True

    load_best_models(request, dataset.lower())

    return dataset_data

def clear_all_models(request, dataset):
    model_types = ["sarima", "bayesian", "winters", "lstm"]

    for model_type in model_types:

```

```

newlist = [model for model in request.session["saved_{0}".format(model_type)]
            if model['dataset'].lower() != dataset.lower()]
request.session["saved_{0}".format(model_type)] = newlist
request.session.modified = True

def load_best_models(request, dataset):
    model_dir = 'static/results/'
    best_models = os.listdir(model_dir)
    for filename in best_models:
        with open("{0}{1}".format(model_dir, filename), 'r') as json_file:
            model_results = json.load(json_file)
            if model_results['dataset'].lower() == dataset:
                save_json_to_session(request, model_results)

def save_json_to_session(request, model_results):
    model_results['id'] = get_timestamp()
    model_type = model_results['model_type']

    request.session["saved_{0}".format(model_type)].append(model_results)
    request.session.modified = True

def get_MSE(actual, predictions):
    total = 0

    for i in range(actual.size):
        total += (actual[i] - predictions[i])**2

    return total / (actual.size - 1)

def get_RMSE(actual, predictions):
    return math.sqrt(get_MSE(actual, predictions))

def get_MAD(actual, predictions):
    total = 0

    for i in range(actual.size):
        total += math.fabs(actual[i] - predictions[i])

    return total / (actual.size)

def get_MAPE(actual, predictions):
    total = 0

    for i in range(actual.size):
        total += math.fabs((actual[i] - predictions[i]) / actual[i])

    return (total / (actual.size)) * 100

def get_timestamp():
    return str((datetime.now() - datetime.utcfromtimestamp(0)).total_seconds() * 1000.0)

```

3. SARIMA Source Code

```

def model_sarima(dataset_data, dataset_name, train_set_idx, my_order, my_seasonal_order,
                 is_boxcox, lmbda):
    r_base = get_r_package('base')
    r_utils = get_r_package('utils')
    r_generics = get_r_package('generics')

    r_stats = get_r_package('stats')
    r_forecast = get_r_package('forecast')
    r_bayesforecast = get_r_package('bayesforecast')

    # Initialization
    test_set_date = dataset_data.iloc[-1]['Date']
    no_of_forecasts = (2050 - (test_set_date.year + 1) + 1) * 4
    forecast_dates = pd.date_range(start=test_set_date, periods=no_of_forecasts, freq="QS")

    train_set_size = train_set_idx
    train_set = dataset_data[0:train_set_size]
    test_set = dataset_data[train_set_size:]

    # -----
    # Using rpy2:
    # Convert inputs
    r_order = objects.FloatVector([my_order[0], my_order[1], my_order[2]])
    r_seasonal_order = objects.FloatVector([my_seasonal_order[0], my_seasonal_order[1],
                                           my_seasonal_order[2]])
    r_null = objects.r['as.null']()

    # import data, train-test split
    r_dataset_data = [float(i) for i in dataset_data['Volume'].values.tolist()]
    r_dataset_data_ts = r_stats.ts(data = r_base.as_numeric(r_dataset_data), frequency =
    4, start = [1987,1])
    r_train_set = r_stats.ts(r_dataset_data_ts[0:train_set_size], frequency = 4, start =
    [1987,1])
    r_test_set = r_stats.ts(r_dataset_data_ts[train_set_size:len(r_dataset_data_ts)],
    frequency = 4, start = [2020,1])

    # boxcox transform
    if is_boxcox and lmbda == 0:
        r_lambda = r_forecast.BoxCox.lambda(r_train_set)
        lmbda = r_lambda[0]
    elif is_boxcox:

```

```

        r_lambda = lmbda
    else:
        r_lambda = r_null

    ## create model
    r_new_model = r_forecast.Arima(r_train_set, r_order, r_seasonal_order, r_null, True, False
        , False, r_lambda, False, "ML")

    # getting fitted values
    r_new_model = r_forecast.Arima(r_dataset_data_ts, r_order, r_seasonal_order, r_null,
        True, False, False, r_lambda, False, "CSS-ML", r_model)
    r_one_step_forecasts = r_stats.fitted(r_new_model)[len(r_train_set):len(
        r_dataset_data_ts)]

    # metrics and forecasts
    r_metrics = r_generics.accuracy(r_one_step_forecasts, r_test_set)
    r_forecasts = r_generics.forecast(r_dataset_data_ts, model=r_model, h=no_of_forecasts)

    # End of rpy2
    # -----

    predictions = []
    forecasts = []

    for x in r_one_step_forecasts:
        predictions.append(x)
    for x in r_forecasts[1]:
        forecasts.append(x)

    # Model Evaluation
    model_MSE = r_metrics[1]*r_metrics[1]
    model_RMSE = r_metrics[1]
    model_MAPE = r_metrics[4]
    model_MAD = get_MAD(test_set['Volume'].values, predictions)
    model_BIC = r_model[15][0]

    forecast_dates = pd.date_range(test_set['Date'][test_set.index.stop-1], periods=
        no_of_forecasts, freq="QS")
    predictions_df = pd.DataFrame(
        {'Date': test_set['Date'],
        'Volume': predictions})
    forecasts_df = pd.DataFrame(
        {'Date': forecast_dates,
        'Volume': forecasts})

    predict_plot = pd.concat([predictions_df, forecasts_df], ignore_index=True)
    graph = get_graph()

    return {
        "graph" : graph,
        "predictions" : predictions,
        "forecasts" : forecasts,
        "test_set" : test_set,
        "bic" : model_BIC,
        "mse" : model_MSE,
        "rmse" : model_RMSE,
        "mape" : model_MAPE,
        "mad" : model_MAD,
        "lmbda" : lmbda,
    }
}

```

4. Bayesian SARIMA Source Code

```

def model_bayesian(dataset_data, dataset_name, train_set_idx, my_order, my_seasonal_order,
    is_boxcox, lmbda):
    r_base = get_r_package('base')
    r_utils = get_r_package('utils')
    r_generics = get_r_package('generics')

    r_stats = get_r_package('stats')
    r_forecast = get_r_package('forecast')
    r_bayesforecast = get_r_package('bayesforecast')

    # Initialization
    test_set_date = dataset_data.iloc[-1]['Date']
    no_of_forecasts = (2050 - (test_set_date.year + 1) + 1) * 4
    forecast_dates = pd.date_range(start=test_set_date, periods=no_of_forecasts, freq="QS
        ")

    train_set_size = train_set_idx
    train_set = dataset_data[0:train_set_size]
    test_set = dataset_data[train_set_size:]

    no_of_iterations = 5000

    # -----
    # Using rpy2:
    # Convert inputs
    r_order = robjects.FloatVector([my_order[0], my_order[1], my_order[2]])
    r_seasonal_order = robjects.FloatVector([my_seasonal_order[0], my_seasonal_order[1],
        my_seasonal_order[2]])
    r_null = robjects.r['as.null']()

    # import data, train-test split
    r_dataset_data = [float(i) for i in dataset_data['Volume'].values.tolist()]
    r_dataset_data_ts = r_stats.ts(data = r_base.as.numeric(r_dataset_data), frequency =
        4, start = [1987,1])

```



```

r_train_set = r_stats.ts(r_dataset_data_ts[0:train_set_size], frequency = 4, start =
[1987,1])
r_test_set = r_stats.ts(r_dataset_data_ts[train_set_size:len(r_dataset_data_ts)],
frequency = 4, start = [2020,1])

# boxcox transform
if is_boxcox and lmbda == 0:
    r_lambda = r_forecast.BoxCox.lambda(r_train_set)
    lmbda = r_lambda[0]
elif is_boxcox:
    r_lambda = lmbda
else:
    r_lambda = r_null
if is_boxcox:
    r_data_transf = r_forecast.BoxCox(r_train_set, r_lambda)

# create model
r_model = r_bayesforecast.stan_sarima(ts=r_data_transf, order=r_order, seasonal=
r_seasonal_order, prior_ar=r_bayesforecast.normal(0,1), prior_ma=r_bayesforecast.
normal(0,1), prior_sigma0=r_bayesforecast.inverse.gamma(0.01,0.01), iter =
no_of_iterations)

# getting fitted values
r_data_fitted = r_generics.forecast(r_data_transf, model=r_model, h=len(test_set))
if is_boxcox:
    r_data_fitted = r_forecast.InvBoxCox(r_data_fitted[1], r_lambda)
else:
    r_data_fitted = r_data_fitted[1]

# metrics and forecasts
r_metrics = r_generics.accuracy(r_data_fitted, r_test_set)
r_forecasts = r_generics.forecast(r_dataset_data_ts, model=r_model, h=no_of_forecasts)

# End of rpy2
# -----

predictions = []
forecasts = []

for x in r_data_fitted:
    predictions.append(x)
for x in r_forecasts[1]:
    forecasts.append(x)

# Model Evaluation
model_MSE = r_metrics[1]*r_metrics[1]
model_RMSE = r_metrics[1]
model_MAPE = r_metrics[4]
model_MAD = get_MAD(test_set['Volume'].values, predictions)

forecast_start = test_set['Date'][test_set.index.stop-1] + pd.DateOffset(months=3)
forecast_dates = pd.date_range(forecast_start, periods=no_of_forecasts, freq="QS")
predictions_df = pd.DataFrame(
    {'Date': test_set['Date'],
     'Volume': predictions})
forecasts_df = pd.DataFrame(
    {'Date': forecast_dates,
     'Volume': forecasts})

predict_plot = pd.concat([predictions_df, forecasts_df], ignore_index=True)
graph = get_graph()

return {
    "graph": graph,
    "predictions": predictions,
    "forecasts": forecasts,
    "test_set": test_set,
    "mse": model_MSE,
    "rmse": model_RMSE,
    "mape": model_MAPE,
    "mad": model_MAD,
    "lmbda": lmbda,
}

```

5. Holt-Winters Source Code

```

def model_winters(dataset_data, dataset_name, train_set_idx, trend, seasonal, damped,
is_boxcox, lmbda):
    # Initialization
    test_set_date = dataset_data.iloc[-1]['Date']
    no_of_forecasts = (2050 - (test_set_date.year + 1) + 1) * 4
    forecast_dates = pd.date_range(start=test_set_date, periods=no_of_forecasts, freq="QS")

    train_set_size = train_set_idx
    train_set = dataset_data[0:train_set_size]
    test_set = dataset_data[train_set_size:]

    # Checking Inputs
    if trend.lower() in ("mul", "multiplicative"):
        trend = "mul"
    else:
        trend = "add"

    if seasonal.lower() in ("mul", "multiplicative"):
        seasonal = "mul"
    else:
        seasonal = "add"

```

```

# Creating Holt-Winters Model
if is_boxcox:
    if lambda == 0:
        lambda = stats.boxcox(dataset_data["Volume"])[1]
    df_data = stats.boxcox(train_set['Volume'], lambda=lambda)
else:
    df_data = train_set['Volume']

model = ExponentialSmoothing(df_data, seasonal_periods=4, trend=trend, damped_trend=
    damped, seasonal=seasonal)
model_fit = model.fit()

# Fitting with test set
predictions = model_fit.forecast(len(test_set))
if is_boxcox:
    predictions = special.inv_boxcox(predictions, lambda)
predictions = pd.Series(predictions, index=test_set.index)

# Predicting future values
forecasts = model_fit.forecast(no_of_forecasts)
if is_boxcox:
    forecasts = special.inv_boxcox(forecasts, lambda)

# Model Evaluation
model_MSE = get_MSE(test_set['Volume'].values, predictions.values)
model_RMSE = get_RMSE(test_set['Volume'].values, predictions.values)
model_MAPE = get_MAPE(test_set['Volume'].values, predictions.values)
model_MAD = get_MAD(test_set['Volume'].values, predictions.values)

forecast_dates = pd.date_range(test_set['Date'][test_set.index.stop-1], periods=
    no_of_forecasts, freq="QS")
predictions_df = pd.DataFrame(
    {'Date': test_set['Date'],
     'Volume': predictions})
forecasts_df = pd.DataFrame(
    {'Date': forecast_dates,
     'Volume': forecasts})

predict_plot = pd.concat([predictions_df, forecasts_df], ignore_index=True)
graph = get_graph()

return {
    "graph" : graph,
    "predictions" : predictions,
    "forecasts" : forecasts,
    "test_set" : test_set,
    "mse" : model_MSE,
    "rmse" : model_RMSE,
    "mape" : model_MAPE,
    "mad" : model_MAD,
    "lambda" : lambda,
}

```

6. LSTM Source Code

```

def model_lstm(dataset_data, dataset_name, train_set_idx, n_inputs, n_epochs, n_units,
    is_boxcox, lambda):
    # Initialization
    activation = 'relu'
    test_set_date = dataset_data.iloc[-1]['Date']
    no_of_forecasts = (2050 - (test_set_date.year + 1) + 1) * 4
    forecast_dates = pd.date_range(start=test_set_date, periods=no_of_forecasts, freq="QS")

    train_set_size = train_set_idx
    train_set = dataset_data[0:train_set_size]
    test_set = dataset_data[train_set_size:]

    # Creating LSTM Model
    lambda = stats.boxcox(dataset_data["Volume"])[1]

    if is_boxcox:
        if lambda == 0:
            lambda = stats.boxcox(dataset_data["Volume"])[1]
        transf_volume = stats.boxcox(train_set['Volume'], lambda=lambda)
        df_data = pd.DataFrame({
            'Date' : train_set['Date'],
            'Volume' : transf_volume,
        })
    else:
        df_data = train_set

    scaler = MinMaxScaler()
    scaler.fit(df_data.set_index('Date'))
    scaled_train = scaler.transform(df_data.set_index('Date'))
    scaled_test = scaler.transform(df_data.set_index('Date'))
    df_data = scaled_train

    generator = TimeseriesGenerator(df_data, df_data, length=n_inputs, batch_size=1)
    model = Sequential()
    model.add(LSTM(n_units, activation=activation, input_shape=(n_inputs, 1)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')

    early_stop = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)
    model.fit(generator, epochs=n_epochs, callbacks=early_stop)

```

```

# Fitting with test set
transf_prediction_results = []
first_eval_batch = df_data[-n_inputs:]
print(type(first_eval_batch))
current_batch = first_eval_batch.reshape((1, n_inputs, 1))

for i in range(len(test_set)):
    current_pred = model.predict(current_batch)[0]
    transf_prediction_results.append(current_pred)
    current_batch = np.append(current_batch[:,1:,:], [[current_pred]], axis=1)

prediction_results = scaler.inverse_transform(transf_prediction_results)
prediction_results = [item[0] for item in prediction_results]

predictions = pd.Series(prediction_results, index = test_set['Date'])
if is_boxcox:
    predictions = special.inv_boxcox(predictions, lambda)
    predictions = pd.Series(predictions)

# Predicting future values
forecast_results = []
first_eval_batch = np.array(transf_prediction_results[-n_inputs:])
print(type(first_eval_batch))
current_batch = first_eval_batch.reshape((1, n_inputs, 1))

for i in range(no_of_forecasts):
    current_pred = model.predict(current_batch)[0]
    forecast_results.append(current_pred)
    current_batch = np.append(current_batch[:,1:,:], [[current_pred]], axis=1)

forecast_results = scaler.inverse_transform(forecast_results)
forecast_results = [item[0] for item in forecast_results]

forecasts = pd.Series(forecast_results)
if is_boxcox:
    forecasts = special.inv_boxcox(forecasts, lambda)

# Model Evaluation
model_MSE = get_MSE(test_set['Volume'].values, predictions.values)
model_RMSE = get_RMSE(test_set['Volume'].values, predictions.values)
model_MAPE = get_MAPE(test_set['Volume'].values, predictions.values)
model_MAD = get_MAD(test_set['Volume'].values, predictions.values)

forecast_dates = pd.date_range(test_set['Date'][test_set.index.stop-1], periods=
    no_of_forecasts, freq="QS")
predictions_df = pd.DataFrame(
    {'Date': test_set['Date'],
     'Volume': predictions.values})
forecasts_df = pd.DataFrame(
    {'Date': forecast_dates,
     'Volume': forecasts})

predict_plot = pd.concat([predictions_df, forecasts_df], ignore_index=True)
graph = get_graph()

predictions = predictions.values.tolist()
forecasts = forecasts.values.tolist()

return {
    "graph" : graph,
    "predictions" : predictions,
    "forecasts" : forecasts,
    "test_set" : test_set,
    "mse" : model_MSE,
    "rmse" : model_RMSE,
    "mape" : model_MAPE,
    "mad" : model_MAD,
    "lambda" : lambda,
}

```

XI. Acknowledgment

I would like to give my utmost gratitude to all that made this research possible, inside and outside the academe:

My family: my mother April, my father June, my sibling AJ, sister Aimee, my nephew Ace, and my cousin Jon. Your unending support for ever since I was born allowed me to get this far, whether it be in care, in cash, in milk tea, in drinks, in late night talks, in bike rides, in everything. Anything that we've done together were what led me to where I am. To my parents: **we've made it.**

My closest friends *in the field*: Julius, Gab, Ivan, Derick, and Fonzie. Thank you for being my save haven throughout college—its ups and downs, whether life was a breeze or a bind. May we all thrive in our ascent through life when we split up on our separate ways. But in all seriousness, you've made the last semester of our predominantly online classes less lonely.

My closest college friends in the track: Will, Raine, Biboy, and Rice, among others. Thank you for making most of the years in UPM more lively even in the simplest things as eating out or commuting home.

My blockmates of BS Computer Science who have been very supportive. Thank you for maintaining the warmth of our relationships despite only seeing each other face-to-face for less than half of our time in college.

My adviser and co-adviser Ms. Perl Gasmen and Dr. Alex Gonzaga for their guidance and suggestions to refine every aspect of the study, from the paper, to the methods, until the final product and the ending of this manuscript.

My research partners Mr. John Riz Bagnol and Mr. Jethro Gem Villoria for their collaboration in perfecting the statistical analysis, as well as their general support both in the study and the conferences that follow.

The university for developing the knowledge and morals in me in order to complete the study, in honor and in excellence.

The country for cultivating in me a sense of love and call to action to perform the best of his abilities for a greater purpose.

Lastly, to God, for allowing all of this to be possible.