Forecasting Project Process Documentation Deliverable 2

March 3rd, 2023

Kevin Taylor, Kelly Du, Nathaniel Ho

IEOR 4574: Forecasting – A Real-World Application

# Contents

[Introduction 1](#_Toc128692097)

[Data 2](#_Toc128692098)

[Data Extraction 2](#_Toc128692099)

[Data Diagnostics 2](#_Toc128692100)

[Data Processing 3](#_Toc128692101)

[Variables 4](#_Toc128692102)

[Pre-Modeling 5](#_Toc128692103)

[Trend/Seasonality 5](#_Toc128692104)

[Seasonal Decomposition 7](#_Toc128692105)

[Modeling 8](#_Toc128692106)

[Results 10](#_Toc128692107)

[Deliverable 1 10](#_Toc128692108)

[Deliverable 2 11](#_Toc128692109)

[SARIMA(0, 1, 1)x(0, 1, 1)52 11](#_Toc128692110)

[SARIMA(1, 2, 1)x(1 , 2, 1)52 Hyperparameter-Tuned 13](#_Toc128692111)

[Facebook Prophet: Chosen Parameters 15](#_Toc128692112)

[Facebook Prophet: Hyperparameter Tuning Per Account 16](#_Toc128692113)

[Results Summary 17](#_Toc128692114)

[Team Information 18](#_Toc128692115)

# Figures

[Figure 1: Project Workflow 1](#_Toc128692201)

[Figure 2: Initial DataFrame Head 2](#_Toc128692202)

[Figure 3: Four Accounts’ Electricity Usage 3](#_Toc128692203)

[Figure 4: MAPE Calculation 5](#_Toc128692204)

[Figure 5: Train-Validation-Test Split 5](#_Toc128692205)

[Figure 6: Average Energy Use (kWh) - All Accounts 6](#_Toc128692206)

[Figure 7: PSD of Average Energy Use 6](#_Toc128692207)

[Figure 8: Weekly Average Use - Seasonal Decomposition 7](#_Toc128692208)

[Figure 9: All Accounts Combined Model Fit / Prediction 10](#_Toc128692209)

[Figure 10: SARIMA(0, 1, 1)x(0, 1, 1)52 Validation Set 11](#_Toc128692210)

[Figure 11: SARIMA(0, 1, 1)x(0, 1, 1)52 Validation Set 12](#_Toc128692211)

[Figure 12: SARIMA Hyperparameter Tuning 13](#_Toc128692212)

[Figure 13: SARIMA(1, 2, 1)x(1, 2, 1)52 Validation Set 13](#_Toc128692213)

[Figure 14: SARIMA(1, 2, 1)x(1, 2, 1)52 Test Set MAPE 14](#_Toc128692214)

[Figure 15: Facebook Prophet (Chosen Parameters) - Validation Set 15](#_Toc128692215)

[Figure 16: Facebook Prophet (Chosen Parameters) Test Set MAPE 15](#_Toc128692216)

[Figure 17: Facebook Prophet (Hyperparameter Tuning per Account) Test Set MAPE 16](#_Toc128692217)

[Figure 18: Team Information 18](#_Toc128692218)

# Introduction

There are many different algorithms can be developed for timeseries such as traditional Extreme Learning Machine (ELM) as well as other commonly used machine learning methods like Recurrent Neural Network (RNN), Linear Regression (LR), k-Smooth Regression (KSR), k-Nearest Neighbor Regression (kNNR), Gaussian Process Regression (GPR), and Generalized Regression Neural Network (GRNN). In this engagement, we are using SARIMA from the *statsmodels* library and Facebook Prophet to forecast future values in a timeseries.

The purpose of this document is to provide a detailed technical overview of:

1. Data
2. Variables
3. Pre-Modeling
4. Modeling
5. Results

Diagram

Description automatically generated

Figure : Project Workflow

The goal of this project is to develop an accurate model for forecasting energy consumption. To do so, the process consisted of reading and pre-processing the data, identifying potential variables, and fitting and testing various models. For Deliverable 1, a SARIMA model was fit to the data for each user account and was tested. For Deliverable 2, the same SARIMA model as Deliverable 1, a SARIMA model with hyperparameters tuned, and FB prophet models with and without hyperparameter tuning were fit to the data.

A timeseries model can create value by predicting energy consumption outside the study period. To develop this model, cleaning the raw dataset was necessary, as was evaluating the accuracy of an initial forecasting model on the data.

# Data

The data was gathered from the archive under the title “LD2011\_2014.txt,” a text file delimited by semicolons. Decimal values were encoded with a comma ie) “0,00.” Values in the dataset were listed in kW power over 15-minute intervals. To convert values to kWh, the values were divided by 4. Missing values, including those for accounts created during the timeframe, are encoded in the data with zeros. The biannual time change results in either one hour of zero readings, or two hours aggregated, depending on the season (March time change vs. October time change).

The python *pandas* package was used to read the data delimited by “;” in to a DataFrame object, followed by some data preprocessing. The first entries are shown below. The accounts pictured either were not in existence at this time or did not record any electricity usage for these timeframes.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Initial DataFrame Head

## Data Extraction

Data extraction was over the entire dataset. The time period of the data is 2011-2014, but some accounts were created during the timeframe – these values are encoded as zeros for the times the accounts were not in existence. After initially processing the datetimes of the data, we are left with a dataset with the shape of (140256, 371), or 140,256 rows and 371 columns. There was one column to record the date/time, and all other columns were readings for 370 accounts.

Each row corresponds to a date and time electricity usage was recorded, while each column corresponds to the reading for that account.

### Data Diagnostics

We checked the quality of the data, including the number of records, duplicates, and missing values. There were no duplicates or missing values to report. It is worth noting that, due to daylight savings time, there are values in March from 1:00am to 2:00am that are zeros, while in October there are values that are averaged between 1:00am to 2:00am. Since this action occurs periodically, we aggregated the data without special treatment for these records**.**

## Data Processing

To pre-process our data into a workable form, we first transformed the data into long pivot form. This helped transform the data so that we could group rows according to year and week. From there, we divided the values for each week by 4; this is because the data collected was in 15-minute intervals. Therefore, to transform a kilowatt power in 15-minute intervals to kilowatt-hours (kWh), dividing by 4 was necessary. We grouped the data by year, week, and account, then summed the values. The first four accounts are visualized below, created from the pre-processed DataFrame. These accounts were not registered for electricity until the year 2012, so there are zero values for the first year.

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure : Four Accounts’ Electricity Usage

# Variables

Weekly usage of electricity is our target variable. It is denoted as the variable *value* within our model. For an individual account, this represents the account’s total usage in kWh for a particular week. To perform initial EDA, we also researched the data’s trend/seasonality on the dataset aggregated over all accounts, by mean weekly usage. The group gained the following understanding about model variables:

|  |  |
| --- | --- |
| *Target Variables* | *Predictive Variables* |
| * Variable that is predicted in the forecasting model. * Weekly usage of electricity in kWh is our “y.” * Denoted in DataFrame as *value* | * Variables that predict target variable * Organized into direct or derived variables   + Direct variable: Directly from dataset   + Derived variable: Created by manipulating direct variables * All variables are direct |

The initial model fit for this deliverable, a SARIMA model, does not have any exogenous (predictive) variables. Rather, SARIMA components use target variables at different lags and weighted average forecast errors as predictor variables, as well as a seasonal component. The Facebook Prophet models we fit did not use exogenous variables either.

# Pre-Modeling

Pre-modeling was performed in two main workflows:

* Exploratory Data Analysis
  + Trend and seasonality in the data are identified for initial modeling. This exercise is performed on all accounts aggregated by week, and the takeaways and model assumptions are then applied to each individual account.
* Pre-Model Utility Functions
  + Utility functions are created for a train-validation-test split, mean absolute percentage error (MAPE) calculation and walk-forward validation, important functions that can and will be applied to future models as well.

A picture containing text, clock

Description automatically generated

Figure : MAPE Calculation

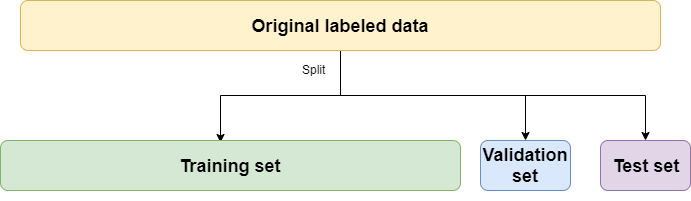


Figure : Train-Validation-Test Split

## Trend/Seasonality

Using the utility functions created to assist the process, the time series and power spectral decomposition (PSD) indicated a first-order trend and a yearly seasonality in the aggregated data. In fact, the maximum power in the PSD corresponded to a period of 51.75 weeks, rounded to 52 weeks or one year.

Chart

Description automatically generated

Figure : Average Energy Use (kWh) - All Accounts

Chart

Description automatically generated

Figure : PSD of Average Energy Use

## Seasonal Decomposition

The process of time series decomposition breaks down a series into its constituent parts: level, trend, seasonality, and noise. This decomposition provides a clear and comprehensive model for analyzing and understanding time series, which is essential for effective time series analysis and forecasting. In the following plot we think of the components as combining additively: y(t) = Level + Trend + Seasonality + Noise with period of 52 weeks.

Performing and visualizing a seasonal decomposition revealed two important points. First, the first-order trend indicates first-order differencing would be required in a model. Second, the seasonal component appears to have a yearly period. For initial model parameters, a SARIMA(0, 1, 1)x(0,1, 1)52 model was chosen. This model was later tuned in its hyperparameters using MAPE as a cross-validation metric, and the model parameters were adapted.

Chart, line chart

Description automatically generated

Figure : Weekly Average Use - Seasonal Decomposition

# Modeling

SARIMA is an extension of ARIMA models that considers seasonality. In a SARIMA model, the hyperparameters that need to be adjusted are the parameters used in regular ARIMA models (p,d,q) as well as additional parameters for seasonality (P,D,Q,s). Thus, we have a total of 7 parameters that need to be optimized.

According to the SARIMA formula (p,d,q)(P,D,Q,s), the parameters for these models are defined as follows:

|  |  |
| --- | --- |
| ***Variable*** | **Explanation** |
| *p and seasonal P* | Represents the number of autoregressive terms (the number of lags in the stationary series). |
| *d and seasonal D* | Represents the number of differences required to make the series stationary. |
| *q and seasonal Q* | Represents the number of moving average terms (the number of lags in the forecast errors). |
| *s* | Represents the length of the seasonal cycle in the data. |

Facebook Prophet is a timeseries regression model able to include a trend component, seasonal components, holidays and other factors. It allows for quick fitting and prediction on timeseries data. Prophet models were used in addition to SARIMAX in the modeling process. Our Facebook Prophet models were created with the following parameters that were tuned for each individual account:

|  |  |
| --- | --- |
| ***Variable*** | **Explanation** |
| *period* | A float representing the period of the FB Prophet model’s yearly seasonality |
| *fourier\_order* | A float corresponding to the complexity of the seasonality. |
| *monthly* | A Boolean (True/False) parameter specifying whether to include a monthly (period = 30.5) seasonality in the model. |

**Evaluation Metric**

The Mean Absolute Percentage Error (MAPE) is a commonly used metric to evaluate the performance of a SARIMA model. This metric is calculated by taking the absolute difference between each predicted data point and its corresponding test point, dividing that value by the test point, and then averaging all absolute percentage differences to give the MAPE. The MAPE formula is shown in the pre-modeling section of this document.

# Results

## Deliverable 1

In Deliverable 1, to evaluate the model effectiveness, we fit a SARIMA model to the data aggregated over all accounts and calculated the MAPE between the predicted values and the actual data. The test set was created from the last 20% of the data and the training set from the first 80% of the data. The test set was not split into multiple sections as it was in Deliverable 2. After fitting the model to the aggregated data, we performed the same modeling for each account. For each account, the test set was created from the latter 20% of the data and the training set from the first 80% of the data. Fitting a SARIMA(0, 1, 1)x(0,1, 1)52 resulted in a test MAPE of **4.5%** on the aggregated dataset.

Chart, line chart

Description automatically generated

Figure : All Accounts Combined Model Fit / Prediction

Performing the same process at the individual account level resulted in many different models, with many different MAPE values. A histogram was developed for the 370 individual accounts’ MAPE calculated on the test set (last 20% of the data). Test set MAPE varied greatly between the individual accounts. The SARIMA model performed very well for some accounts and poorly for others. The Total MAPE was used to aggregate and quantify error for each account, defined as:

Using this MAPE formula, the total test set MAPE was **15.6%**. However, this value is likely skewed by actual values that are very close to zero, resulting in large prediction error percentages. If we instead use the median instead of the mean, median MAPE was **6.5%**.

## Deliverable 2

For Deliverable 2, the data was divided into training (first 60%), validation (following 20%) and test (last 20%) sets. Furthermore, the test set was divided into three equal parts. In the results figures below, these are shown as Test Sets 1, 2 and 3. This division was performed on the aggregated data, and at the individual account level. Hyperparameter tuning for the SARIMA model was performed first on the overall (aggregated) dataset, then the chosen hyperparameters were applied at the individual account level. Hyperparameter tuning was not performed at the individual account level for SARIMA due to the long computation time. For 370 accounts and around 30 seconds to fit a SARIMA model, combined with 27 (3x3x3) combinations of hyperparameters, this would take 80+ hours and was not feasible. However, account-specific hyperparameter tuning was performed with the Facebook Prophet model, which takes a fraction of the time to fit.

At the individual account level, MAPE values were represented with a boxplot and a report of the median MAPE for each test set.

### SARIMA(0, 1, 1)x(0, 1, 1)52

Initially, as part of Deliverable 1, a SARIMA(0, 1, 1)x(0, 1, 1)52 was fit to the overall data, then to each individual account. When fit on the validation set from the overall data, this model had a MAPE of 12.8%.

Chart, histogram

Description automatically generated

Figure : SARIMA(0, 1, 1)x(0, 1, 1)52 Validation Set

When applied to the individual account level, this model had good results and MAPE did not decompose greatly over the test set periods.

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Set** | Test Set 1 | Test Set 2 | Test Set 3 |
| **Median MAPE** | 6.5% | 7.0% | 7.6% |

Figure : SARIMA(0, 1, 1)x(0, 1, 1)52 Validation Set

Overall, this model performed well. The median MAPE on each test set was relatively low, and did not increase greatly between test sets 1, 2, and 3.

### SARIMA(1, 2, 1)x(1 , 2, 1)52 Hyperparameter-Tuned

After fitting the initial SARIMA(0, 1, 1)x(0, 1, 1)52 at the overall and account level, the hyperparameters of the model were tuned to achieve the smallest MAPE on the validation set for the overall dataset.

Text

Description automatically generated

Figure : SARIMA Hyperparameter Tuning

The minimizing parameters of (1, 2, 1) resulted in a validation set MAPE of 10.4%.

Chart, line chart

Description automatically generated

Figure : SARIMA(1, 2, 1)x(1, 2, 1)52 Validation Set

Similar to the previous SARIMA model, this was applied to the individual account level and had varying results. For each test set, the boxplots of account MAPE are shown.

Chart, box and whisker chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Set** | Test Set 1 | Test Set 2 | Test Set 3 |
| **Median MAPE** | 21.1% | 54.8% | 90.6% |

Figure : SARIMA(1, 2, 1)x(1, 2, 1)52  Test Set MAPE

This model did not perform well. This is likely due to the differencing parameter of 2 – a second-order trend applied to most accounts likely caused the prediction to miss the actual value by a great degree in the test set, a problem that was exacerbated further into the future. Performing hyperparameter tuning on the overall aggregated dataset, then applying the same parameters to the individual accounts, was not helpful. However, this method was implemented due to the computational time of SARIMA modeling and overall was not very successful. Using a simpler model with lower order for differencing and no autoregressive component performed much better on the test set.

### Facebook Prophet: Chosen Parameters

A Facebook Prophet model was fit to the overall data with chosen parameters *period*=365.25, *fourier\_order*=12, and *monthly*=True. When fit on the validation set from the overall data, this model had a MAPE of 18.9%.

Chart

Description automatically generated

Figure : Facebook Prophet (Chosen Parameters) -Validation Set

This Prophet model was then applied to the individual account level with the same parameters and had decent results.

Chart, box and whisker chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Set** | Test Set 1 | Test Set 2 | Test Set 3 |
| **Median MAPE** | 15.4% | 14.8% | 17.4% |

Figure : Facebook Prophet (Chosen Parameters) Test Set MAPE

### Facebook Prophet: Hyperparameter Tuning Per Account

The Prophet model was on the hyperparameters *period*, *fourier\_order*, and *monthly* for each account, to find the combination of parameters to minimize each account’s validation MAPE. When this was applied on the test set for each account, the results were similar to the previous Prophet model.

Chart, box and whisker chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Set** | Test Set 1 | Test Set 2 | Test Set 3 |
| **Median MAPE** | 16.0% | 14.9% | 18.1% |

Figure : Facebook Prophet (Hyperparameter Tuning per Account) Test Set MAPE

It seems hyperparameter tuning at the account level did not make a large difference on test error. In fact, test error seems to have increased by a small amount between the two Facebook Prophet models.

## Results Summary

The results for all models tried are shown in the summary table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Aggregated Data Cross-Validation MAPE** | **Test Set 1 MAPE** | **Test Set 2 MAPE** | **Test Set 3 MAPE** |
| **SARIMA**  **(0, 1, 1)x(0, 1, 1)52** | 12.8% | 6.5% | 7.0% | 7.6% |
| **SARIMA**  **(1, 2, 1)x(1, 2, 1)52** | 10.4% | 21.1% | 54.8% | 90.6% |
| **FB Prophet:**  **Chosen Parameters** | 18.9% | 15.4% | 14.8% | 17.4% |
| **FB Prophet: Hyperparameters Tuned by Account** | N/A | 16.0% | 14.9% | 18.1% |

## Future Ideas

For future analysis on this dataset, our team would suggest the following:

* **Incorporation of Exogenous Variables to Transform SARIMA -> SARIMAX:** Since the SARIMA model performed the best on the test set, continuing with this model would be wise. Creation of time-based features such as holiday indicators, or other exogenous variables such as average monthly temperature, could help improve model performance.
* **Other Models:** Trying other models on the data, including neural networks, could improve the performance of the model on the test set.

# Team Information

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Kevin Taylor** | **Nathaniel Ho** | **Kelly Du** |
| MS Data Science Student | MS Data Science Student | MS Data Science Student |

Figure : Team Information