**Predicting Energy Demand**

Group 3: Matthew Fritscher, Ryan Supple, Gabriel Orosco

ISDS 7075 – Business Forecasting

Dr. Young Chun

3 December 2022

# Introduction

The production of energy is extremely important to everyday life and often there is some controversy around the production of energy. How should we produce it? What resources should be used? No matter the answers to those questions everyone agrees that energy needs to be readily available and at a reasonable cost. However, due to storage limitations, energy can’t be created and then stored for when it’s ready to use. This requires energy companies and plants to produce adequate amounts of energy for consumption while minimizing cost and waste. This is where Energy Forecasting has grown in importance. In order to determine how much energy is required, energy companies create forecasts to guide the plants on how much to produce for a given time frame.

# Framing The Problem

The energy forecasts need to be as accurate as possible in order to reduce the cost, waste, and resources needed. In order to create accurate forecasts many factors can be taken into account, such as weather conditions, calendar dates, economic activity, and electricity prices. However, for our forecasts we can only utilize past records of energy loads, temperature readings, and calendar dates. The data we have is hourly readings or temperature and energy load for each day from the beginning of 2015 to 2019. The temperature readings are not from the same location as the energy plants, but it is assumed they are in the same region. Our goal in our forecasting is to provide an accurate prediction that minimizes the root mean square error value in order to minimize the costs and waste produced in energy production.

# Data Overview

We gathered two datasets for our forecasting. The first dataset was of temperature readings. This dataset contained hourly temperature readings in Fahrenheit from nine different weather stations throughout the region with no info on location or relevance to the energy zones. The second data set contained hourly energy production in fifteen different energy production zones in the region. However, the energy data has some missing time periods within the data. There are eight additional weeks in the past that need to ‘back-casted’ as well as the one week ahead forecast prediction to make.

One of the first steps in creating the forecast is making sure the dataset is accurate and clean. To do so we started by attempting to identify outliers and other possible errors in the data. In the temperature dataset we identified a few outliers. For example, we found a recorded temperature drop of 40 degrees in a single hour followed by another jump of 40 degrees the next hour. To address these outliers, we took the average temperature from the last two accurate readings and replaced the outlier values. In the energy dataset, we discovered that some dates included in the dataset were days that never happened. For example, in the set were two leap year dates of February 29th in years that were not leap years. These dates were identified and removed from the dataset. With the data cleaned, we were able to start exploring data.

# Data Exploration

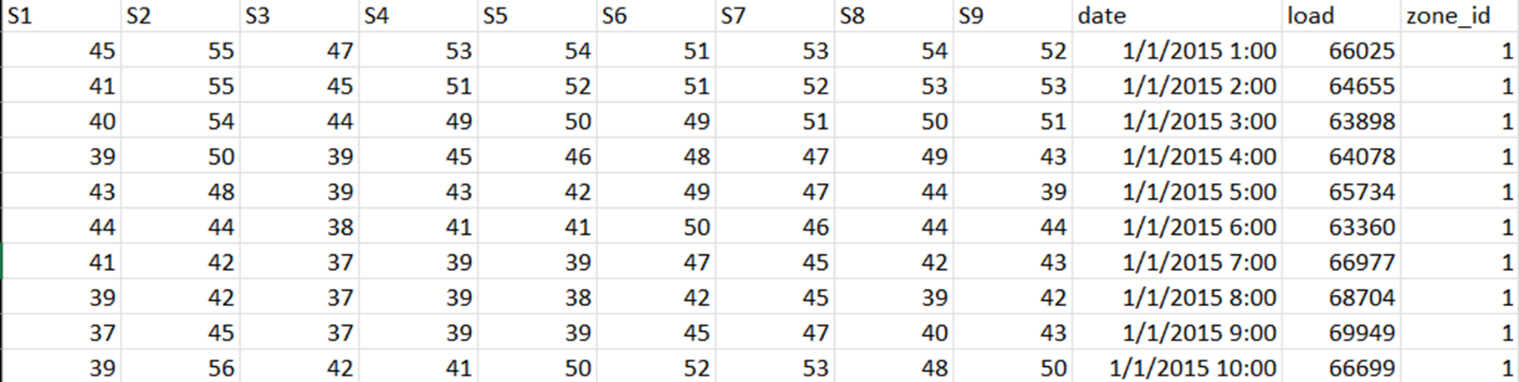
In order to create accurate forecasts, we needed to understand the data and identify certain trends and other possible variables that can be affecting the data. In looking at the data two seasonal trends easily stood out. The first one was the yearly trend and change in energy consumption. As the weather changes for each season energy consumption may change. For example, in the summer and winter months cooling and heating units could see heavy usage and increase the overall demand for a region. The second trend seen in the data is the daily energy consumption. Since our data contains records for every hour during a day, the time of day could have an influence on energy consumption. The amount of energy consumed during the night hours may be less than during the busy day hours. These two seasonal trends allow us to understand what kind of models will work better with our data.

In exploring the data, we also identified a couple of variables that could influence energy consumption in a given region. The first of these variables was determining if the given date was a holiday, like Christmas or the Fourth of July. These are days that can see a change in energy consumption for each zone as people are not in their usual daily schedule and possibly are doing unique activities on these days. The other variable we decided could be influencing the energy loads was determining if it is the weekend or not. On weekends people vary from the other five days of the work week and thus office buildings, downtowns, and other places that are consuming energy heavily during those days may require less energy on weekends. To address both of these variables we added two columns to our forecasting dataset. These columns were both binary variables that identified if the date fell on a holiday or weekend. By adding the extra binary variables to the dataset, it allowed us to make better predictions and better explain the forecasted values.

# Data Pre-Processing

Once we had gathered the data, understood the data, and prepared the data we were ready to take the next steps. This meant getting the data in the right format for usage before trying to create forecasting models in Excel, Python, Splunk, etc.

Given that temperature data included information from 15 different weather stations, and the energy data included information from 9 zones, initially there was no feasible method to correlate which stations corresponded with which zones. The reason there was no way to correlate the temperatures with the loads was due to the fact that the matrices were of different dimensions (15 x 24 and 9 x 24). The solution to this problem was to pivot the energy data so that the hourly loads were stored as rows instead of columns. With the energy data transposed, it was possible to join the energy and temperature tables using an outer join. This format allowed the use of the 9 temperature variables as independent variables in some of the various prediction models developed. Figure 1 shows an example of the transposed data frame. Wrangling the data like so allowed us to easily run and test different models to determine the best possible forecasts.



Methods: Prediction Models

## Naïve Models:

Various models were created to create predictions for energy demand. Naïve models were built to provide a baseline understanding of what an acceptable RMSE score should be. A seasonal naïve model used the same values for energy for the missing days as the previous year. For example, if a back-cast for March 6-10, 2016, were to be conducted, then the values from March 6-10, 2015, were to be used in this model. This model evaluated to have an extremely high RMSE of over 115,000.

Additionally, a Seasonal Average Naïve model was used, and provided much better results than the first naïve model. The predictions from this model improved the RMSE for the back-cast predictions (but not for the forecasts for June 2019). This model used the past and future to fill in the back-casts. For example, if a back-cast for March 6-10, 2016, were to be conducted, then the values from March 6-10, 2015, and March 6-10, 2017, were averaged and used as the predicted values.

Ultimately, the naïve models provided a benchmark from where the predicted values were to be improved.

## Regression Model:

Another model we tried was a linear regression model in python. The linear regression model utilized all the variables we had created. It included the two dummy variables for weekend and holiday as well as each temperature station’s recordings for the day. The goal in creating this model was to attempt to identify the correlations and relationships between the different variables and to see if it produced accurate forecasts. However, with our understanding of advanced models we knew a simple linear regression model would not suffice at predicting a seasonal trend like the one present in this dataset. This resulted in our RMSE value being around 16000. Meaning each of our predictions were off by an average value of 16000. This model proved too inaccurate for our liking and we moved onto a try different models. If we were to try and improve our linear model, we would have started with normalizing the temperature variables and assuring we utilized a quadratic regression equation.

## ARIMA Model:

The next model we implemented was a seasonal ARIMA model. This model seemed like the perfect choice as the model is supposed to handle the data de-seasonalized and it utilizes more recent records to calculate new predictions moving forward with moving average. However, we were ultimately unimpressed with this model as well. When we tested the model we produced in python, using the pmdarima package, it gave us a RMSE value of 10000. This still seemed high when we were trying to make accurate forecasts and thus, we decided to continue to search for better models. If we were to revisit this model, then I think we may have utilized a different package to build the model or test other versions of the model. The package we used identified the best ARIMA model to be a (1,0,0) by using AIC value, meaning it was just a first order autoregressive model. We aren’t convinced that using the AIC value was the best way to determine the best ARIMA model.

## Long Short-Term Memory

### Predicting Temperatures

The Long Short-Term Memory model (LSTM) was used to predict the temperatures. A LSTM model is a type of Neural Network. Using the TensorFlow package in Python, the temperatures for each hour of every station were trained in two different ways. One model only considered the temperature from the previous day at a specific area, i.e., hour 1 data was only considered to train hour 1 data from other days. The model was inserted into a function that was called for each of the 24 temperature “variables”. This model implied that the temperature for the future is only affected by the temperature from the previous days at a specific hour. In reality, it may be useful to use all 24 hours to train the model. For instance, cold fronts often come in the evening; using the aforementioned model would not be able to use the data from the evening in order to predict the next morning’s temperature. For this reason, another LSTM model was developed such that all 24 hours of temperature data were considered. Ultimately, this model yielded better results. The temperature predictions from this model were used to predict the energy demand for the forecast for 6/23-6/29, 2019.

### Predicting Energy Demand

With success using the LSTM model to predict temperatures, a similar LSTM model was developed to predict energy demand. This model used temperature and energy data to predict energy demand. The results from this model were less accurate than one of the Naïve models, and therefore, was ultimately not considered to use to make the final predictions.

Challenges that were encountered when using the LSTM to predict energy demand included over-estimation of low-demand areas and under-estimation of high-demand areas. This was likely due to the fact that zone IDs were not considered as dummy variables (research indicated that including dummy variables into this sort of neural network is dubious at best). This LSTM model had difficulties learning which zones were typically low or high demand zones. Perhaps another strategy could be to separate the time series by station and train and fit the model, develop predictions, insert the predictions into a data-frame, and integrate all predictions back into the original format. Additionally, this model was difficult to tune the hyperparameters, given the limited documentation on the modules used.

# LLP5 Model

Splunk was used to build a prediction model and do data preparation. The data was originally two separated datasets with no known relationship. As both datasets had some similarities, we made some data transformation in both datasets to merge them and improve the chances of forecasting and back casting attempts opposing the previous models using them as two separate datasets.

After the data cleaning, the data transformation was based on transforming the columns of the 24 hours into rows and merging the fields day, month, and year as a singular field date that we also added the number of hours represented by each row of the transformed 24 hours columns. That allowed both datasets to have a similar key, date, to be blended in. One additional step to this data transformation was to temporarily convert the hour 24 to 0 so as not to overlap the days for the model.

The blend of the two datasets required another extra step, since our focus was to predict the load of energy based on the temperature measured by the 9 weather stations, and their relationship to each of the energy zones were unknown, we decided to transform the rows of each weather station as a column causing an intentional one-to-many relationship in the blend of the datasets, since that way we could make a model that would correlate every energy zone to all of the stations and define the best weight for each.

The final model used in Splunk was LLP5 which combines LLT (Local level trend) and LLP (Seasonal local level) models for its prediction. In other words, this model computes two predictions, one using LLT and the other using LLP. The algorithm then takes a weighted average of the two values and outputs that as the prediction. The confidence interval is also based on a weighted average of the variances of LLT and LLP. The confidence interval was set to 95% both for the lower end and the higher end.

Explaining a bit further, the LLP, which is a univariate model with seasonality, needs a period which has the follows the condition that the number of data points must be at least twice the number of periods. The algorithm takes into account the cyclical regularity of the data, if it exists and if the period is not specified the algorithm tries to calculate it. LLP will return an error message if the data is not periodic.

LLT is a univariate model with trend but no seasonality and requires at least 3 data points.

The process applied to identify the seasonality of the data was made in two steps, first we calculated a timeseries to verify the relevance of each energy zone per day in relation to the total for the same day during the entire time interval of the studied data set, second, we run the LLP model without setting a period so the algorithm would return which period it considers the best. However, this algorithm tends to overestimate the prediction, so we took a weighted average on both periods to run the LLP5.

After transforming the data and analyzing the seasonality the LLP5 model was run isolating the energy zones one by one for each of the nine weeks that needs to be predicted, meaning that the model was run 135 times. During the back cast, the results were a bit better than the results of the forecast. The speculation of that is due to the change of behavior in 2019 for some of the energy zones while the temperature behavior followed the same pattern.

# Conclusion and Summary

### Surprises:

The most difficult problem encountered throughout the project was figuring out an appropriate way to join temperature data with the energy data. The process of data pivoting proved to be the most difficult aspect of the energy forecasting process. However, once the data was transformed into the desired format, it was much easier to join the temperature data with the energy data, which ultimately led to the development of better predictive models.

Ultimately, the LLP5 model was used to generate predictions for the missing energy demand data. With the LLP5 model essentially being a combination of a linear regression and a seasonal forecast, this model is less sophisticated than some of the other models tested, while scoring lower when using RMSE as a metric. Therefore, the parsimonious model reigned superior over the advanced models in this project.

As the LLP5 model was run without the dummy variables for weekend and holidays requires more research on this end to figure out how those two variables could interact with the prediction.

# References

https://docs.splunk.com/Documentation/Splunk/9.0.2/SearchReference/Predict​

​https://www.tensorflow.org/api\_docs/python/tf/keras/layers/LSTM