CS 391L

2023/04/27

Final Project Report

Group Members

Name: Christopher Lawson, EID: cal4555
Name: William Recktenwald, EID: wjr636

Introduction

In this project we apply machine learning to the domain of electric load forecasting. Electric load forecasting is critical for short, medium, and long term optimal control and operation of electric power systems. Load forecasting models can also be applied in the areas of utility capacity planning in regulated territory, Independent System Operator (ISO) planning in non-regulated territory, and may be a key part of energy systems analysis for a diverse set of applications.

Techniques within the fields of statistics and machine learning have been applied to this problem for decades with varying degrees of success. With the proliferation of computing power and improvements in machine learning techniques more accurate and complex predictions have been produced by various practitioners. We survey the state of the applied machine learning techniques currently being applied in the space, apply these models, build upon these models to create our best load forecasting model for a selected dataset, benchmark performance of all applied models, and review theoretical underpinnings of the applied models in order to understand why our best candidate model performs as it does.

Electric load forecasting accuracy is important both on average and at the extremes. Usually models will be successful at interpolation but may fail at the task of interpolation. Notable examples include the 2021 winter storm Uri. ERCOT's load forecast was too low. Such criticism is easy in retrospect, however at such low temperatures only a limited dataset was available for the ISO's load forecasting exports to understand how electricity demand for space heating and other uses would perform in Texas. We focus on average accuracy metrics in our work but acknowledge that examination of extreme cases such as this would be an additional benchmark to validate any proposed electric load forecasting model or model framework.

Data Sourcing

For this project we needed to source both endogenous (electric load) and exogenous (various, primarily climate) data. Electric load was sourced from the Electricity and Reliability Commission of Texas (ERCOT, https://www.ercot.com/gridinfo/load/load_hist). Our primary source of exogenous data has been climate data from the National Oceanic and Atmospheric Administration (NOAA, https://www.ncei.noaa.gov/cdo-web/datatools/lcd).

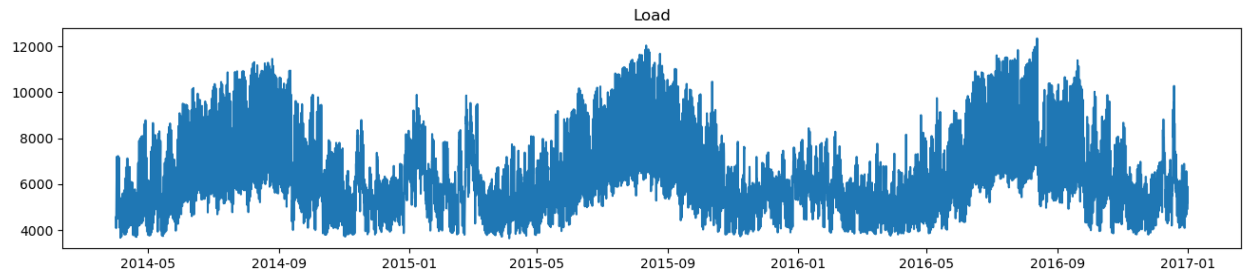
ERCOT data covers electric load for several regions in the State of Texas. NOAA data reports numerous climate data fields for many weather stations in the United States of America with decades of history. In this case we only pulled data for select stations in Texas over the last ten years.

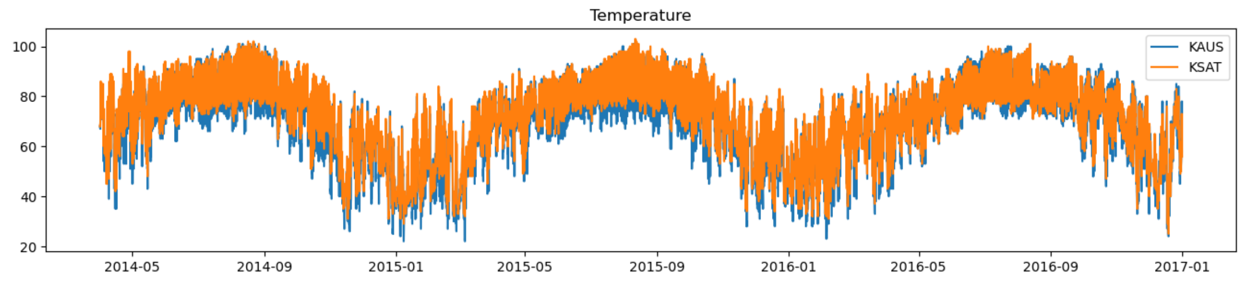
In both cases multiple files needed to be downloaded, parsed and combined in order to build a dataset for training our models. Both datasets were available at hourly granularity.

Data

After sourcing all of our data, we obtained 6 different datasets that represent the 6 different load zones that ERCOT provides electricity for. These zones being: Coast, Far West, North Central, North, South Central and South. Each of these zones has hour by hour information about energy demand along with the area's temperature and humidity. One limitation to our data sourcing was that we could only get this weather data from airports in the given zones. Thus we may be missing out on the accurate weather of the region. In addition to this, we were only able to collect the data of one airport per zone (except for south central). As such we acknowledge that there is more data that we could have trained on. Going forward, we will be continuing with the South Central dataset as this is the zone that contains Austin.

Electric Load for the South Central region, temperatures for the two airport datasets (KAUS: Austin, KSAT: San Antonio) we referenced are plotted below. Notably, this data shows highest load during the hot summer months and is generally low and stable during the Winter period.





Models

Due to time constraints, we were only able to test out a limited set of models. Those models were multiple linear regression models and different iterations of a transformer. Next, because we were curious how the State-Of-The-Art time-series analysis models would do, we also included META's *Prophet* model to compare results. For all models we utilize a test train split of 10 percent.

Linear

We decided to implement a simple linear model to try to have a base model to compare against whenever training different models. Firstly, we would just try to predict our energy demand purely based on temperatures at weather stations to see what kind of results we would get. Next we would complicate it a bit by also including humidity, and the timeseries element into the data. We did so by splitting the datetime into its various components and included that in the data to train on. These various components of date, time were also encoded into binary interaction terms so that the effect of exogenous regressors could interact with the seasonality of the data. Both models were based on multiple linear regression without regularization. We will distinguish these two types of linear models by either having a simple linear model or a complex linear model.

Transformer

The next model we included was a transformer. This model was selected due to its incredible ability to interpret underlying patterns in time series data. Thus given how well it was able to perform on NLP data, we figured we would get a lot of good results. One thing that we needed to do before we could go ahead and train our models, we needed to establish a sliding window to train on along with a sliding window to try to predict for. The values we decided to hypertune on were as follows:

- Training sliding window: [24, 48] (in hours)
- Evaluation window: 24 (hours).

Next, we decided upon a single transformer encoder architecture for our model to see if we could get any kind of good results before we would try to attach multiple encoder and decoder layers and over complicate our model. With this, we hypertuned on 3 more variables: number of encoder layers, number of heads for multihead attention, and a percentage of dropout throughout the encoder layer.

The following are the values hypertuned over for the transformer specific parameters:

- Number of encoder layers: [1, 4, 8]
- Number of heads: [1, 5]
- Dropout Percentage: [0%, 10%, 20%]

After this transformer layer, we also attach a single linear layer to condense the results into our single target value, the electric load demand.

An extra feature that we hypertuned over for the transformer model was the learning rate and the number of epochs trained over. The following were the values we tried:

- Learning rate: [1e-3, 1e-4]
- Max Epochs: [10, 20]

META - Prophet

The last model we wanted to implement is META's *Prophet*. This model was developed by META to try to solve their own time-series data. This model attempts to break down the given data into the following equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Where the g(t) is the trend function which models non-periodic changes, s(t) is the periodic changes in the values, and h(t) represents the effects of holidays. These holidays are defined as special time periods where the captured event may not be behaving as normal. For example, if we were to try to model UT foot traffic during the year, we may want to include a special "holiday" parameter to account for anytime UT has a home game as that increases the average foot traffic. Lastly, ϵ_t is an error term which is assumed to be normally distributed.

Fortunately, we were able to use their open source libraries to import their model to see how well it would fit to our data. The only two things that we tested with this model was whether we provided all of the data to the model (temperature, humidity, etc) or only provided the model with the time series component along with the demanded electric load.

Results

Multi-Linear Regression

The following table contains the results for the MLR models.

Complexity	**Train/Test**	**RMSE**
Simple	Train	1410.454
Simple	Test	1464.745
Complex	Train	308.5117
Complex	Test	319.0298

Transformer

The following are the top 5 models along with their hypertuned values:

Encoder Layers	Number of Heads	Dropout	Learning Rate	Max Epochs	Training Window	Evaluation Window	RMSE
1	1	0.0	1e-3	10	24	24	1706.727

1	1	0.0	1e-2	10	24	24	1706.993
4	1	0.0	1e-2	10	24	24	1707.050
4	1	0.2	1e-4	10	24	24	1707.324
8	1	0.0	1e-4	10	24	24	1707.378

META - Prophet

Figures 1 through 4 plot the model fit to all the data provided to each model where the blue line is the model fit and the black points are the data points. Figures 1 and 3 plot the training splits, Figures 2 and 4 plot the testing splits. Figures 1 and 2 showcase the model only being trained on timeseries information and Figures 3 and 4 showcase the model being trained on all information.

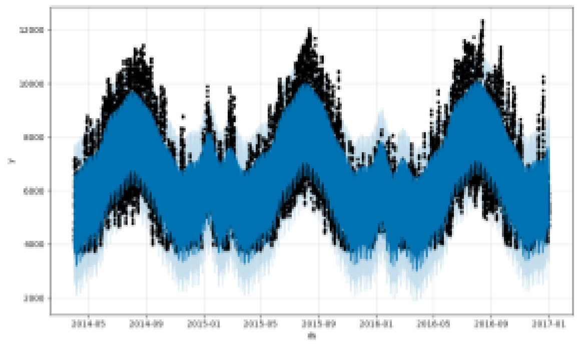


Figure 1 - Prophet Training on only timeseries information

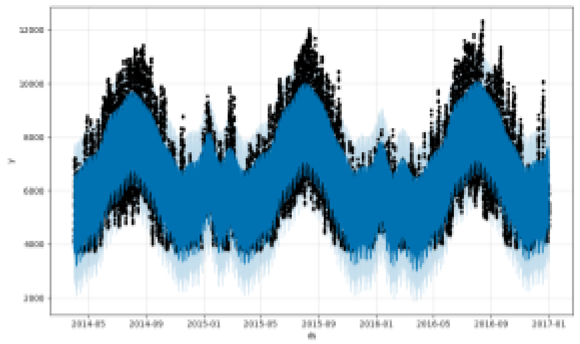


Figure 3: Prophet Training on all information

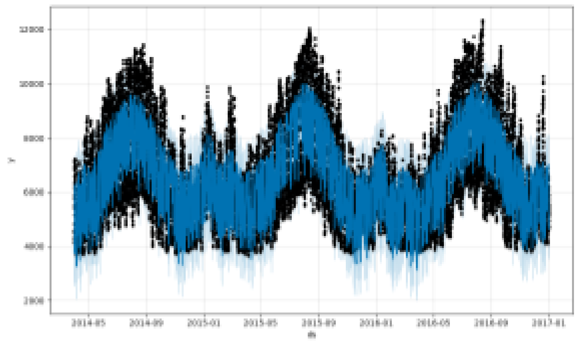


Figure 2: Prophet Testing on only timeseries information

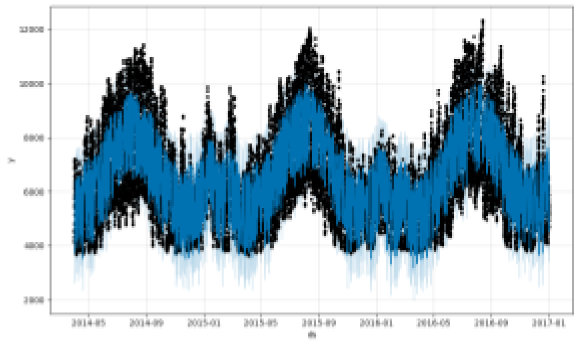


Figure 4: Prophet testing on all information

Table 1 contains the RMSE values for the 4 figures.

Data Trained on	**Train/Test**	**RMSE**
Timeseries	Train	2150.635

Timeseries	Test	2220.895
All	Train	2150.140
All	Test	2225.565

Tabel 1: Prophet Results

Discussion

Our findings are indicative of the challenges in applying machine learning to real world applications. Machine learning models are large, complex systems capable of being tuned and utilized in diverse domains. In order to succeed large, well organized datasets, as well as expertise within machine learning, and possibly subject matter expertise, must be combined. Results are dependent on the interaction between each of these factors and each machine learning model built for application in the real world.

Measured in terms of RMSE, our most successful model was the complex Multiple Linear Regression (MLR) with time series based interaction terms. The dataset was the same size for all models so this is not a differentiator, however some models may be more demanding in terms of how much data is available for training. In this case, the interaction terms do severely limit the dataset by effectively splitting it across several periods. However, in the context of MLR there is an efficient method for setting a bias, updates to the coefficients and bias occur with more data but technically a model can at least “learn” an average from a single data point. This means that while we only had several 8,760 hour annual seasonality periods this was likely enough the MLR model to learn some level of bias and coefficient accuracy for all interaction types.

This limited set of 8,760 hour annual seasonality periods likely is what limited our transformer models. While we could have increased the window beyond the 24 to 48 hours we tested many, many more years of load and weather data would have been required in order to have the full seasonality window of 8,760 hours repeated enough times to train a model. In this case we believe that the limited size of our dataset as compared to its seasonality drove the best of the various transformers to underperform the best of MLR as measured by RMSE.

What was more surprising than the comparative performance of transformer models to MLR was the performance of the META Prophet model. This model is billed as state of the art for time series forecasting, and is frequently referenced in application by practitioners in the space. After using this model and finding worse performance than the transformer, we note that prophet is probably best applied to problems with limited exogenous regressors. In this case, we have a clear relationship between several exogenous regressors and our endogenous data. Including the exogenous regressors is demonstrated as more valuable than modeling the time series seasonality relationship by the simple MLR model outperforming META prophet.

Conclusion

Electric load forecasting is an incredibly important problem to try to tackle. This helps local infrastructure to be able to prepare in advance for demands of electric load along with allowing the structures in place to be flexible in where generated power is diverted and distributed. Given this, we were able to generate a plethora of data to try to train models. To try to expand upon our models capabilities we incorporated more indicative data (namely weather information) to see if we could further help our models in predicting energy load.

The resulting best model was the complex Multiple Linear Regression which resulted in a test RMSE of 319. This implies that, while it was a time series problem, as long as we provided the correct information into the MLR model we would be able to predict the correct predicted load primarily relying on exogenous regressors.

Repository

All code and data used within this report are contained within this repository:
https://github.com/cLawson101/CS-391L-Final-Project

Files of interest are noted in the README file of the repository.