Expanding Upon Short Term Energy Forecasting Using Various Models

Jake Kosakoff,* Nicole Liu,† Lydia Nizheborsky,‡ and William Ross§
University of California, San Diego

(Group 10)

(Dated: June 18, 2024)

This paper attempts to develop and improve upon the energy-load-predicting neural network model proposed in *Short Term Energy Forecasting with Neural Networks* [1]. In addition to using updated and current data from the Federal Energy Regulatory Commission, PJM (an East Coast electricity merchant), and the National Oceanic and Atmospheric Administration, we analyze five prediction models to achieve minimal Mean Absolute Percent Error (MAPE), thus determining which model is optimal for maximum forecast accuracy. The five models include two forms of linear regression, a boosted decision tree, a neural network, as well as a neural network with added boosting. We also utilize the principle of interacting variables and nonlinearities discussed in the paper. We find that the best predictions were made using the neural network with additional boosting through residuals, with an average MAPE of 1.39% across 100 trials.

I. INTRODUCTION

Electricity usage, generated through various means, is increasingly more common and critical in the world. Our reliance on it continues to grow, thus it becomes increasingly important that we have a reliable and consistent method to deliver this energy when needed. Part of this process involves ensuring the electricity grid can deliver the requested amount of electricity, the load, at the appropriate time. The requested load is forecasted through load amounts from the previous day, as well as other variables such as weather and day-of-week. This load prediction is then used to maintain and produce the load and electricity for the next day.

It is critical to be able to accurately predict and forecast the load for the next day to avoid outages and other failures. Traditional methods of prediction involve regression and time series analysis [1]; a relatively newer approach involves the use of neural networks to predict the load for the next day. It is this method that we will focus on in this paper. We will follow the models and methods outlined in [1] to develop and train our neural network on contemporary data.

In addition to a neural network, we will also use linear regression and boosted decision trees in our prediction. The linear regressions will connect temperature and load as well as attempt to predict how the entire set of data predicts the load. We will also enact this process using a boosted decision tree to predict the load given a set of variables. This boosted decision tree will also be used to boost the residuals of the trained neural network to further refine and predict the load. Finally, we will analyze our results between methods and compare trends to [1].

II. DATASET

The data used in this paper was pulled from various energy companies and government organizations. Data about energy load and various calendar variables (day-of-week, holidays, etc.) was pulled from the Federal Energy Regulatory Commission and the East Coast electricity merchant PJM [2, 3]. Additional data about weather and temperature variables was pulled from the National Oceanic and Atmospheric Administration [4]. There are a total of 1095 entries in our data with 29 calendar and weather variables. Our models will use variations upon this data to predict the load.

The same data is also presented in different formats depending on the model. Calendar variables for the boosted decision tree can be written as 1-12 representing months, for example. For regression and neural networks, this data is re-aggregated into 12 columns, with 1 or 0 representing whether that data entry occurred that month.

Additionally, we will implement two forms of regression in this paper. One of them includes the engineered data and interaction variables while one does not. These interactions were created using the initial dataset in the same method as the paper [1]. Each model is created using one training and testing split and then further tested using 100 other training and testing splits for cross validation purposes.

III. METHODS

This project will use four main methods of prediction to predict the load given some data. The four methods are linear regression, boosted decision trees, neural networks, and neural networks with boosting.

* jkosakof@ucsd.edu
† n5liu@ucsd.edu
‡ lnizhebo@ucsd.edu
§ wross@ucsd.edu

A. Linear Regression

Our first form of prediction comes through the form of linear regression. We carry out linear regression on various selections of data according to the general equation:

$$H(\vec{x}) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_d x_d \tag{1}$$

where $\vec{x}$ is the input, ω_i is some weight parameter, $H(x)$ is the predicted ouput, and d is the dimension of the input. We carry out three linear regression models in this project.

The first model relates temperature and load based on the quadratic pattern between the two in the data. As a result, we model the temperature-load relationship with

$$H_1(\vec{x}) = \omega_0 + \omega_1 \phi_1 + \omega_2 \phi_2 \tag{2}$$

where $\phi_1 = x$, $\phi_2 = x^2$, and x is the temperature data. Therefore, Eq.2 becomes $H_1(\vec{x}) = \omega_0 + \omega_1 x + \omega_2 x^2$. Running a linear least squares regression on the temperature data using this model to obtain the weights (see Table I) and forecast load results in a MAPE (between predicted and actual load) of 4.452%.

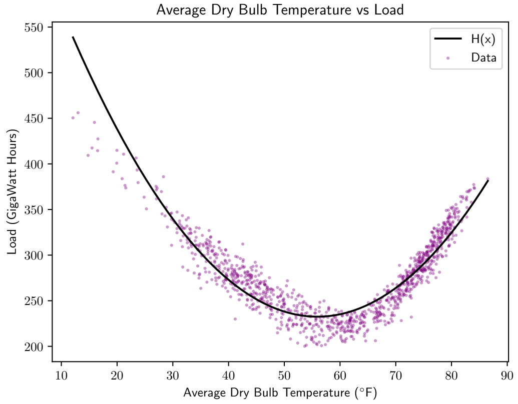

FIG. 1. Load predicted by temperature according to the least squares model $H_1(\vec{x}) = \omega_0 + \omega_1 x + \omega_2 x^2$.

We also engage with multi-variable linear regression. The two approaches taken in this process use the full range of available data and differentiate in whether or not interaction variables are taken into account [1].

In both cases, we follow a least squares linear regression similar to Eq. 1. Using interaction variables and comparing the predicted load against the actual load reveals the following trend in Fig. 2. The average MAPE across 100 trials of different training and testing data splits is 1.51% (see Fig. 8).

Creating a linear regression model without taking into account interaction variables and repeating another 100 trials of training and testing data splits has an average MAPE of 2.96% (see Fig. 9).

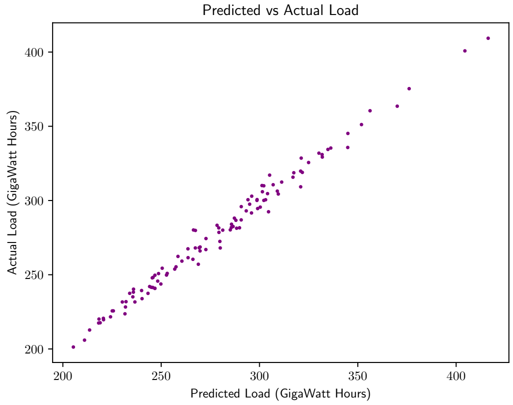

FIG. 2. Predicted against actual load, using a least squares linear regression with interaction variables.

B. Boosted Decision Trees

The next prediction model we incorporate is a boosted decision tree. Training an initial model with suggested base parameters (see Table II) produces a model with a MAPE of 2.076%. We can further optimize these parameters through cross-validation (Table II) to produce a model with a MAPE of 1.508% on the same training and testing dataset. Fig. 3 shows how the optimized predictions, on average, lie closer to the true value.

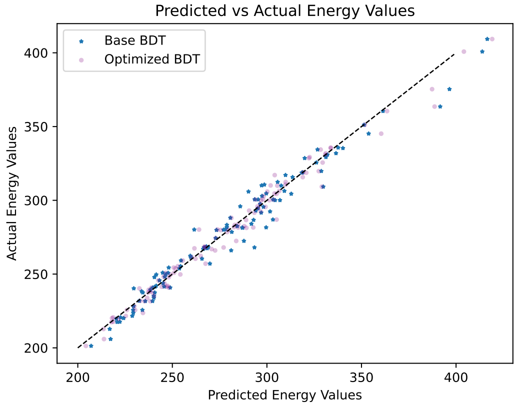

FIG. 3. Predicted load using a boosted decision tree, before and after the parameters for the boosted decision tree have been optimized through cross-validation.

We can further extend these optimized parameters to other iterations of training and testing data splits to ensure that the optimized parameters are not specific to some initial configuration of training and testing data. Training another 100 boosted decision trees on new iterations of training and testing data returns an average MAPE of 1.63%.

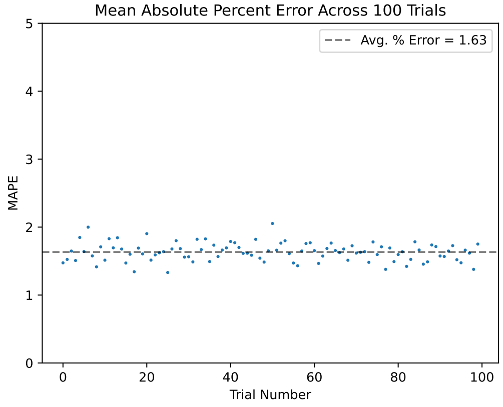

FIG. 4. MAPEs across 100 trials of different training and testing data splits for the boosted decision tree using the optimized parameters detailed in Table II.

C. Neural Networks

Our first attempt at a neural network was created with just one Dense layer, 4 nodes, and sigmoid activation. This network has all 27 variables entering the first layer and results in a single number output, the energy prediction. The training data for this neural network was the same as the boosted decision tree and needed to be linearly converted to a scalar for the activation functions. To do this, we used sklearn's StandardScalar and MinMaxScalar functions [5]. After this model was trained, it resulted in around a 2% MAPE value; however, this model was very simple and we recognized areas for improvement.

Firstly, our model was treating all the data it received in the same way, even though there were two different forms of data: calendar data and weather data. With weather data, we expect complex interactions between the variables. For example, wind on a hot day should decrease the amount of energy needed, whereas wind on a cold day should increase the energy. On the other hand, we expected that the calendar variables would have a much more linear effect on the data. Weekends should have less energy load requirements since offices use significant amounts of energy and most people do not work in an office on weekends.

With this understanding of the data, we rebuilt the model to apply a sigmoid activation function to the weather variables and a linear activation to the calendar variables. This model is shown in Fig. 10. After refactoring our model, we trained it again on the same data, resulting in an improved MAPE of 1.56%. Additional nodes or layers were found to overfit the data and therefore were not used.

D. Neural Networks with Boosting

To find our boosted residuals we put the energy values and calendar/weather variables into our neural network. The output from this gave us our predicted energy values. We then determined the error by subtracting our predictions from the real energy values from the data.

$$\text{error} = E_{\text{actual}} - E_{\text{predicted}} \tag{3}$$

This error then became our new input for our boosted decision tree along with our calendar/weather variables. The output from this is our predicted error values. Adding these error values to our predicted energy values gives us our closest prediction of the energy values. We choose to use predicted error values as they might account for any overarching trend in the otherwise presumed random error. Using predicted error values lowers our MAPE from 1.56% to 1.39%.

The predicted error accounts for any trends as mentioned earlier or common behaviors in error value and thus provides a closer approximation of the predicted values. This was to maximize our forecast accuracy in the hopes that an error predicted from our real errors would better account for any hidden behaviors in our error values.

Further explanations and descriptions of prediction models and methods can be found on our GitHub repository [6].

IV. RESULTS

To talk about the results of our models, we have to look at their MAPE scores. For each MAPE value, we first perform 100-fold cross-validation and average the accuracy across these trials. We can then compare these MAPE values to assess which model is the best. The results are shown in Fig. 5.

We expected the smart regression, normal linear regression with intelligently engineered data and interaction variables, to perform incredibly well. This model had a MAPE of 1.51%, meaning that the predictions made by this model were only 1.51% off of the actual expended energy values on average. This was a significant improvement over the linear regression without engineered fields (MAPE of 2.96%). With this high benchmark set, our goal was to improve upon this value using only the unprocessed data set.

Boosted decision trees and neural networks were both able to approach the smart regression benchmark, with MAPE values of 1.63% and 1.56% respectively. However, even with optimal parameters, neither could beat this benchmark. This result is in line with [1].

To improve upon these methods, we used a boosted decision tree to find patterns in the residuals from the neural network and used a combination of these methods

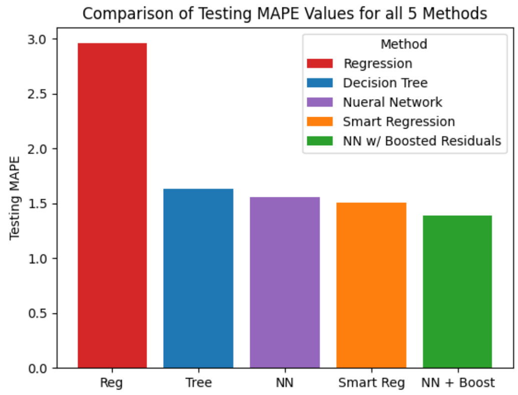

FIG. 5. The average MAPE for each prediction model across 100 trials.

to predict the final energy output. Shown in Fig. 6 are the actual versus predicted residuals of the neural network.

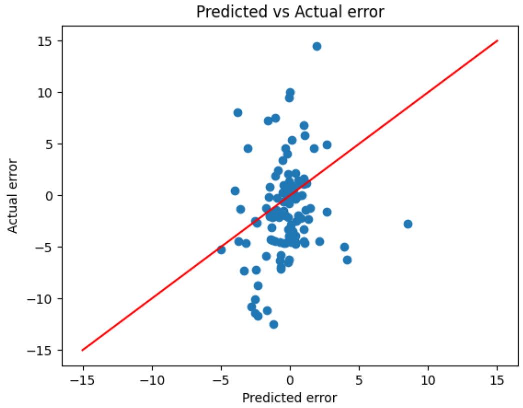

FIG. 6. Actual and predicted errors of the neural network using the boosted decision tree. The red line represents a one-to-one ratio, idealizing a perfect prediction.

The general linear trend of this data showed that the boosted decision tree was able to find a pattern in the error of the neural net. This suggested that using these methods in conjunction with one another should further increase the accuracy of our predictions. This prediction was correct and we can see in Fig 7that our MAPE value decreased.

After this modest yet important improvement, our model now had a MAPE of 1.39%, beating the 1.51% of the smart regression. Ultimately, our combination of machine learning techniques was able to outperform the intelligently engineered data fields and interaction variables used for linear regression. We were able to showcase the strengths of neural networks for tasks such as energy forecasting.

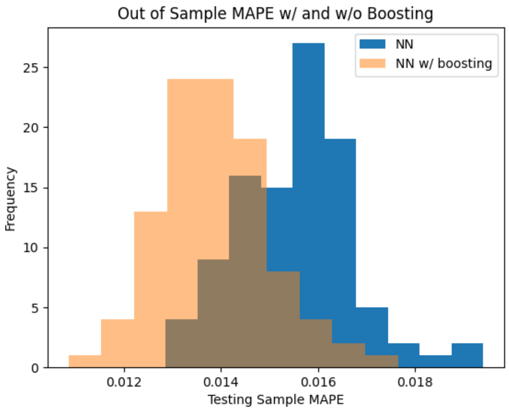

FIG. 7. Actual and predicted errors of the neural network using the boosted decision tree. The red line represents a one-to-one ratio, idealizing a perfect prediction.

V. CONCLUSION

Ultimately, we were able to succeed in our goals to use boosted residuals to improve upon the work done in the paper to predict energy loads. Using boosting on top of a neural network is something that is rare in machine learning, despite its ability to incrementally improve the performance of models. While the improvement in performance (0.12%) may not seem like a lot, getting every ounce of accuracy in a task as specialized as this one is important. By combining two different algorithms, neural networks and boosted decision trees, that analyze a dataset in a different way, you are able to extract weaker patterns that are overpowered by the dominant features of a dataset. As we strive to perfect the prediction abilities of machine learning algorithms, it is vital that more research goes into ways of extracting the subtle interactions that one method by itself cannot pick up on.

This project taught us the importance of building a model one step at a time. When we tried to jump straight into a more complex neural network model, we failed to create any useful predictions. To see why this was happening, we went back to basics and started as simple as possible, a linear regression with only the temperature variable, and then slowly incorporated more and more variables until we had good results. Then we extended this process to decision trees and neural networks, again slowly building up from the most simple model possible. In the future, it would be good to continue trying to increase the model complexity by adding in more hand picked variable interactions, as well as potentially trying multiple rounds of boosting to reduce the error even further.

VI. CONTRIBUTIONS

We distributed the work on this project fairly evenly. Jake worked on preprocessing the data, the neural network model, and integrating the neural network with the boosting. Nicole did the regression analysis and optimized the decision tree hyperparameters. Lydia worked on generating plots to describe the dataset and creating the decision tree framework. Will set up the neural network and worked on building the neural network up piece by piece.

Appendix A: Plots

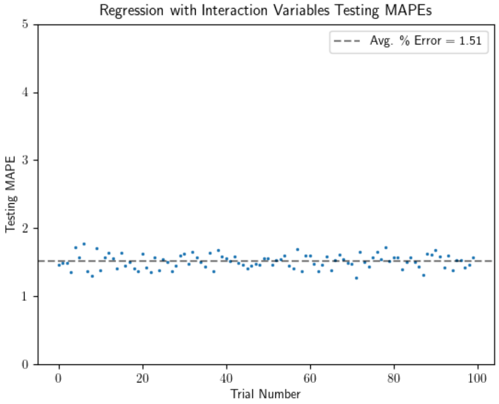

FIG. 8. MAPEs across 100 trials of linear regression having taken into account interaction variables.

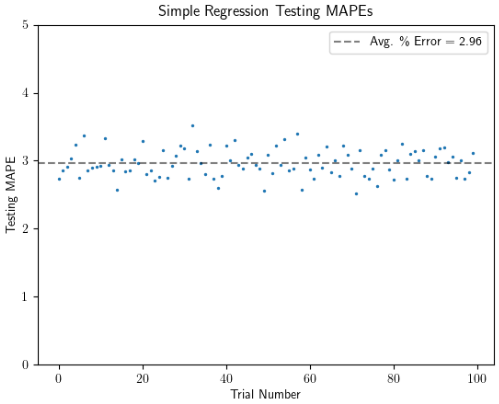

FIG. 9. MAPEs across 100 trials of linear regression without taking into account interaction variables.

Appendix B: Tables

Weight	Value
ω_0	730.783
ω_1	-17.816
ω_2	0.159

TABLE I. Weights for $H_1(\vec{x})$ (Table I).

	Initial	Optimized
n_estimators	100	1000
max_depth	3	5
learning_rate	0.1	0.1
colsample_bytree	0.8	0.6
subsample	0.8	0.7

TABLE II. Initial and optimized parameters for the boosted decision tree.

Appendix C: Additional material

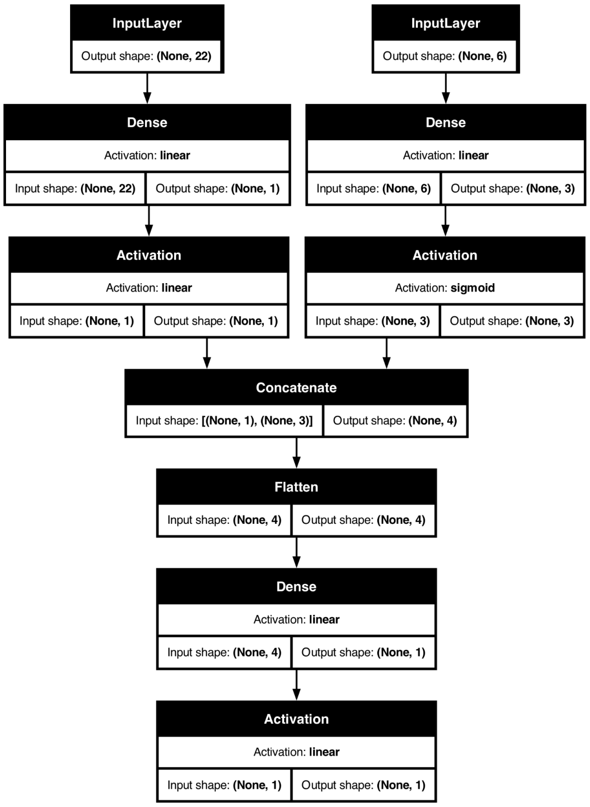

FIG. 10. The neural network model and architecture.

[1] J. Stuart McMenamin and F. A. Monforte, Short term energy forecasting with neural networks, The Energy Journal **19**, 43 (1998).

[2] Federal energy regulatory commission (2024).

[3] Pennsylvania-new jersey-maryland interconnection (pjm) (2024).

[4] National oceanic and atmospheric administration (2024).

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research **12**, 2825 (2011).

[6] J. Kosakoff, N. Liu, L. Nizheborsky, and W. Ross, jakekosakoff/physics-139-final (2024).