Solar Power Generation Forecasting Using a Random Forest Regressor

Parker Nelms
Instructor: Dr. Lei Yang
CS 458 Introduction to Data Mining
13 December 2022

Abstract

Solar generation forecasting is an important part of providing renewable energy and optimizing the power outputs across a network of solar plants. In this paper, I propose a simple machine-learning approach to solar power forecasting using a random forest regressor trained with time-series data. I use 24 hours of historical data to get a short-term prediction with a forecasting horizon of 24 hours. By using grid-search optimization, feature reduction, and data splitting, I was able to achieve results that would allow the three given solar plants to predict solar output a day ahead with fairly good accuracy.

Introduction

Solar power is an abundant and renewable resource that can be used to generate power responsibly. However, the power output of a solar generation plant is highly dependent on environmental factors such as cloud cover, wind, precipitation, humidity, time of year, and much more. For example, solar power output is going to be lower on a cloudy winter day, than on a sunny summer day. It is critical that a solar grid operator can anticipate how much solar power it will be able to generate at any given time in the future, whether it be for managing the load on a power grid or for changing power prices. Additionally, data that produces good forecasting accuracy is easy to obtain using common sensors, so developing a model to analyze that data is important for solar power plants across the world.

Solar power generation through the use of machine learning models has been studied thoroughly in the last few decades as a viable solution. Consequently, many models have been proposed with varying success. Different learning models are more useful for some solar forecasting applications than others. Specifically, different models work best for short-term (up to 6 hours ahead), medium-term (up to 7 days ahead), and long-term (weeks to months ahead) forecasting [1]. It is important to note that I did come across many different definitions for short and medium-term forecasting, however, for the purposes of this paper, I will use the previously given definitions.

The problem presented in the project description is to create a solar power generation forecasting model that can accurately predict up to 24 hours ahead on a rolling basis. The data provided contains two years of hourly measurements on certain environmental variables. The data is further broken down into three different solar power sites which means that each zone will have different environmental data on any given hour or day. Two different approaches can be taken with this zone data when training a model. First, the model can be trained using all training data from all three sites. Alternatively, the data can be split into different zones and a

model can be trained using each. This would give a distinct model that each zone can use to predict the solar output.

Regardless of how many models are used, the result of the predictions will be evaluated using two different metrics. The first is the mean absolute error (MAE) and the second is the root mean squared error (RMSE). The formulas for each are given here:

$$MAE = \frac{1}{number\ of\ points}\sum_{t} |p_t - \hat{p}_t|$$

$$RMSE = \sqrt{\frac{1}{number\ of\ points}\sum_{t} |p_t - \hat{p}_t|^2}$$

Both the RMSE and MAE basically just give the average distance between predictions and ground truth. However, RMSE gives more weight to larger errors and is thus more sensitive to outliers. For this reason, it can be expected that the RMSE will be higher than the MAE score. Also, since the given power outputs are normalized between 0 and 1, the error scores should, in turn, be between 0 and 1, with a lower score being better.

Background and Related Work

Many different models have been used in the past to predict medium-term solar power generation. The naive approach for a 24-h ahead model is to take the current solar output and predict it as the output 24 hours in the future. While this approach is simple, by using more complex models, it quickly becomes easy to see that great improvements can be made using regression.

Previous studies have been done with similar data and problems. Rangjian et al. claim that XGB is useful for prediction where historical data is lacking [4]. They used data that included sunshine data, temperature data, and routinely available geographical information for same-day prediction. The study shows that with good data, even a semi-naive approach to medium-term forecasting is viable. Their RMSE was around 21%, which is fairly good for this type of problem.

Using a sparse CNN, Feng et al. achieve a maximum RMSE of around 7% and a minimum of around 3.6%. They use sky images to predict solar generation [2]. I consider this a much more modern and expensive solution, however, it does produce far better results than many of the other approaches listed here.

Paoli et al. used an MLP regression model to predict on a 24-hour horizon using 19 years of daily GHI data. They were able to improve a minimum MAE and RMSE score of 9.40% and 12.43% with an ANN model, to an average MAE and RMSE score of 8.83% and 11.96%, respectively, using the MLP model [5]. The GHI is the geometric sum of the Direct Normal Irradiance (DNI) and Diffuse Horizontal Irradiance (DHI) [6].

Technical Approach

For my technical approach, I will explain my approach to data preprocessing, which model I selected and why I selected it, and my approach to hyperparameter optimization. My data preprocessing was a four-step process that included feature reduction, zone splitting, time series conversion, and normalization.

In order to reduce features in my data, I started by creating a correlation matrix so that I could visualize how each feature correlates with the power output and each other. As you can see in figure 2, variables 78, 79, and 134 had a low correlation with the power output. These features can be removed with little to no effect on error scores. I can also get rid of either variable 169 or 178 because they have a nearly one-to-one correlation with each other. This means that only one will be useful for predicting the target. Since variable 169 has a slightly lower correlation with power, I removed that feature. Removing these features will become very important when we train the model since the time-series data will end up having many features and reducing those as much as possible will reduce the training time.

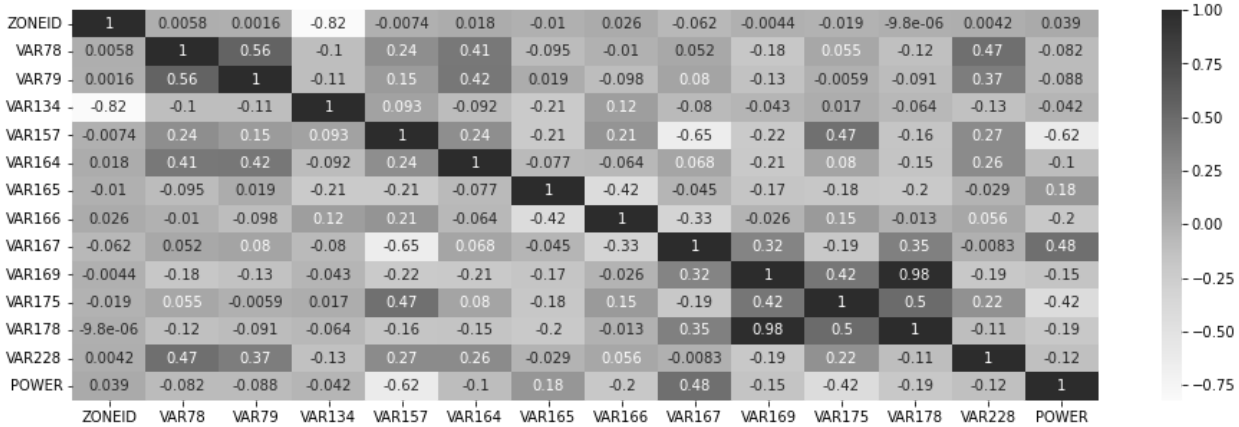

Figure 1: Correlation matrix of all features in the dataset before feature reduction.

Next, I decided to split the data into separate datasets for each zone ID for several reasons. The first reason is that I was testing mostly with one zone since it reduced my training time with each new change I made. Another reason was that, since the data for each zone is more representative of its own specific environment, I knew that the testing accuracy would most likely be better in the end, even if the training set was smaller. In other words, the training data that I lost would likely not be representative of the true test values anyway. Also at this step, I converted the timestamp objects for each sample into a month, day, and hour value that could be interpretable by the regressor.

After each zone was split into different datasets, they were converted to time series datasets. This was done by combining/concatenating the features of n samples where n is the number of hours of historical data that is used in prediction. For example, if we consider 24 hours of historical data and if each sample contains 8 features of useful data (as was decided in this model's case) then one sample of the time series data would contain 8*24 = 192 features. Now, we can say that the target power value for that sample is the power output value 24 hours after the last historical point is considered. The final time series dataset will be the length of the original zone dataset minus the first 23 hours, for which we can not predict the 24-hour ahead power value without 24 hours of historical data, and minus the last 24 hours in which there are no target values available.

The last step of my data preprocessing stage is normalization. In order to get valid results and normalized error score ratings, I needed to make sure that the feature data was normalized between 0 and 1 as well. I did this by using a simple normalization formula on each zone dataframe, df:

$$df_{new} = \frac{df - df_{min}}{df_{max} - df_{min}}$$

Now that all of my data was ready to be read by a regression model, I needed to select the correct model. Based on the previous research done and my knowledge of models that are useful for this task, I decided to train a random forest regressor, an MLP regressor, a support vector regressor (SVR), and a stacking regressor (using MLP, SVR, and Gradient Boost). By judging the initial training results without optimization, I confirmed that the random forest regressor handled the data and predictions the best. After optimization, I was able to get results similar to the random forest with the MLP regressor, however, the random forest was still better with my best hyperparameter optimizations.

To perform hyperparameter optimization, I used a grid search to find the best combination of parameters. Interestingly, I tried the hyperparameters found in "Hyper‐parametric improved machine learning models for solar radiation forecasting" [3] with

my random forest model and achieved better or the same error score and with a much lower training time. I ultimately ended up using those parameters with small changes. See table 1 for the table of hyperparameters used in the paper.

Table 1: The grid-search results for a similar application of a random forest regressor that was used in my implementation [3].

Parameter	Grid Search	Random Search	Bayesian Optimization	Description
DHI	E – 30	E – 30	E – 98	*N* estimators
	D – 20	D – 30	D – 21	Maximum depth
	F – sqrt	F – sqrt (square)	F – 3	Maximum features
	MSL – 2	MSL – 4	MSL – 1	Minimum sample leaf
DNI	E – 30	E – 30	E – 100	*N* estimators
	D – 20	D – 20	D – 46	Maximum depth
	F – 0.5	F – sqrt (square)	F – 4	Maximum features
	MSL – 2	MSL – 4	MSL – 1	Minimum sample leaf

In order to test the models, I had to perform the same preprocessing steps I used on the training data with the testing data. Once I got my list of predictions for the test set, I used the sklearn.metrics library to get the MAE and RMSE scores. I also compared the predicted values with the true values by plotting a day in each season for all three zones and plotting the values against each other so I could visualize how close the predicted values fit to the true values for every power level.

Evaluation Results

Since I import the data directly from the ".csv" files, testing with a new dataset is as simple as replacing the "solar_test.csv" file with your new data when running the full program. To make verification easier, the day plots give exact time stamps so that the true value can be verified. I also created a test script at the end of the notebook that can be used to output the exact true value and prediction at any given time. To run change this script to check for a specific value, just change the i value and the number on the y_test and pred variables as explained in the comments of the code. You can then cross-check the true value/timestamp with the solar_test file and day comparison line plots.

The results of my model are pretty consistent with an average MAE score of 5.966% and an average RMSE score of 11.635%. The exact error scores are laid out in table 2:

Table 2: MAE and RMSE scores for each zone.

ZONEID	1	2	3	Total
MAE	0.05475	0.06205	0.06218	0.05966
RMSE	0.11053	0.11933	0.11919	0.11635

The error score is reflected when comparing the actual true values to the predicted value on any given day (Figure 2). In these graphs, I show the power value comparisons for a full 24 hours in the summer, fall, winter, and spring for each zone. Notably, it shows that the predicted power values follow the true values well, but are most accurate on lower targets while undershooting most of the time on higher targets. Figure 3 also shows that lower power values tend to fit to the true values better than higher ones, especially when compared to another model like the MLP regressor. I suspect the reason for this is that making sure the power is 0 between a certain time of the day is an easy task for a decision tree/random forest.

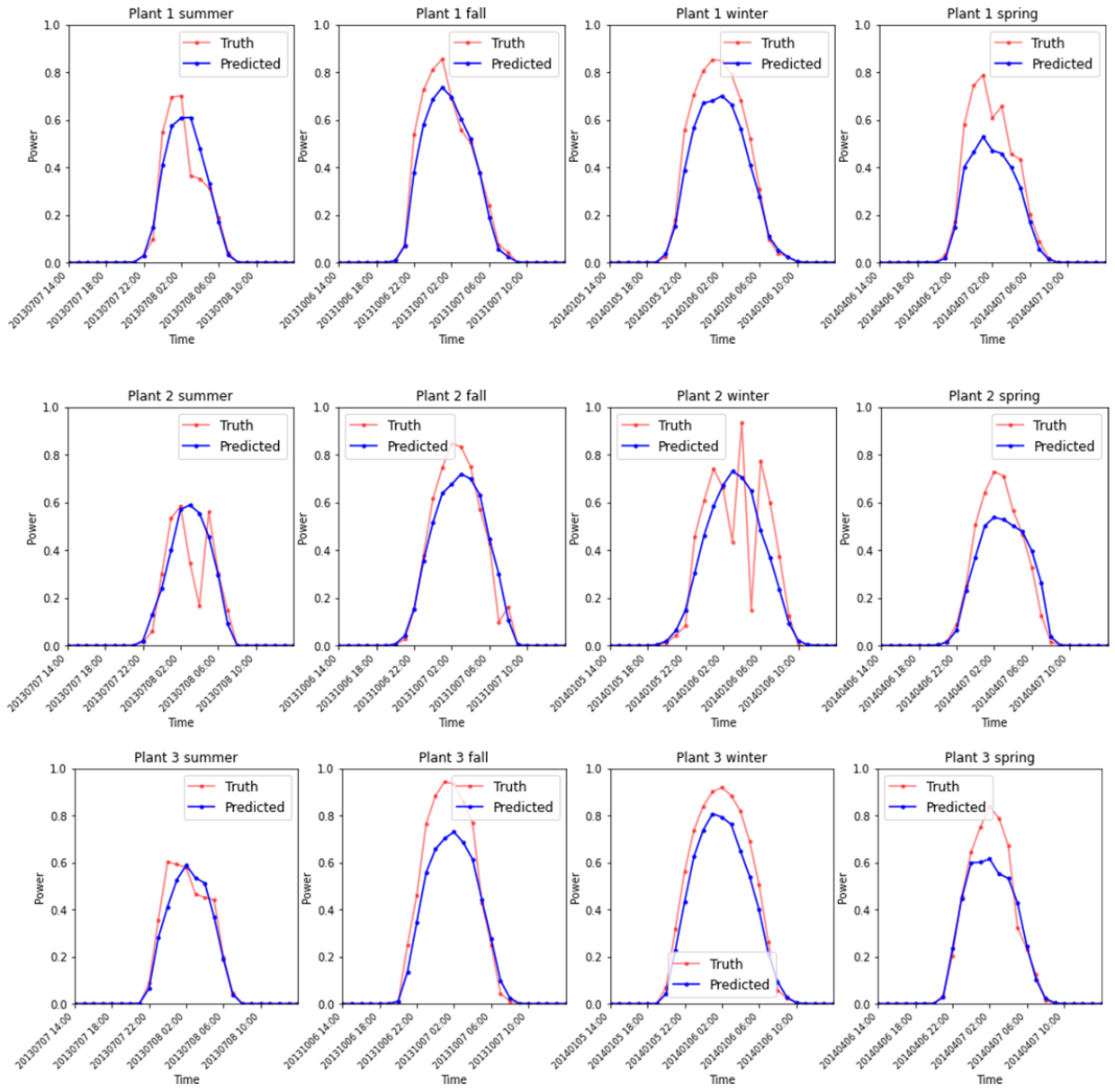

Figure 2: Day comparisons of true vs. predicted power values in different seasons for each zone in a line graph format.

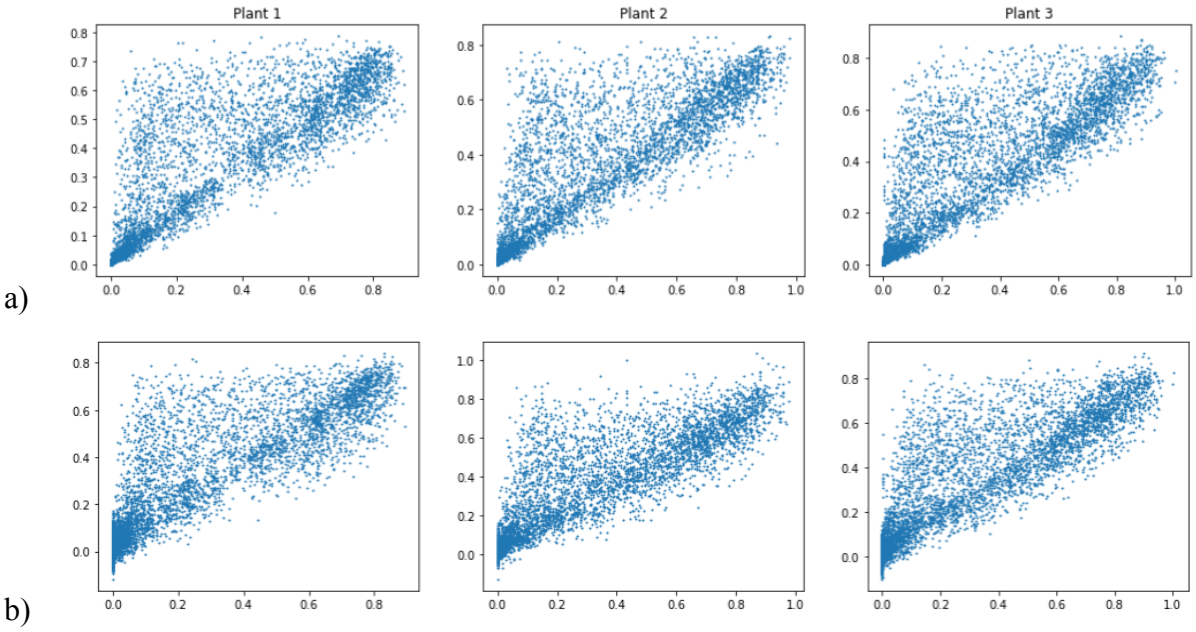

Figure 3: Scatter plot comparing the true and predicted power values for the random forest regressor (a) and the MLP regressor (b).

The benefits of using this model are immediately obvious. The MAE and RMSE scores are very low when compared to the other models in which I generally could not get a lower MAE score and RMSE score of 7% and 12%, respectively, for any of the zones. I also found that I only got an error increase of ~1-2% when I considered just a couple of hours of historical data. This means that the model is still viable with limited features. The runtime when predicting with a random forest is also small compared to other methods since traversing a decision tree is generally quick.

The downsides of this model are that training takes much longer when compared to the other methods and optimizing using grid search is time intensive since training takes so long. However, the time it takes for each of these is negligible when considering that each of these tasks only has to be done once. Another downside is that the model predicts relatively low values when the ground truth is high. One way I might be able to fix this is by finding a model that overshoots the values and taking the average between the two.

Conclusion

The goal of this project was to create a solar power generation forecasting model with as low of an error as possible. Using a random forest regression model, I was able to achieve a fairly low MAE and RMSE score, while also minimizing my training time through feature reduction and hyperparameter optimization. I also found that using 24 hours of historical data maximized my accuracy while not requiring too much computational power. Overall, the final results are satisfactory for this type of problem and with this given dataset, however, better results can be achieved through modern methods such as sky image analysis via CNN.

Resources

1. Perez, R., Kivalov, S., Schlemmer, J., Hemker, K., Renné, D., & Hoff, T. E. (2010). Validation of short and medium term operational solar radiation forecasts in the US. *Solar Energy*, *84*(12), 2161–2172. https://doi.org/10.1016/j.solener.2010.08.014
2. Feng, Zhang, J., Zhang, W., & Hodge, B.-M. (2022). Convolutional neural networks for intra-hour solar forecasting based on sky image sequences. Applied Energy, 310(C), 118438–. https://doi.org/10.1016/j.apenergy.2021.118438
3. Kumar, Namrata, K., & Kumari, N. (2022). Hyper‐parametric improved machine learning models for solar radiation forecasting. *Concurrency and Computation*, *34*(23). https://doi.org/10.1002/cpe.7190
4. Qiu, Rang jian & Liu, Chunwei & Cui, Ningbo & Gao, Yang & Li, Longan & Wu, Zongjun & Shouzheng, Jiang & Hu, Meng. (2022). Generalized Extreme Gradient Boosting model for predicting daily global solar radiation for locations without historical data. Energy Conversion and Management. 258. 115488. https://doi.org/10.1016/j.enconman.2022.115488
5. Paoli, C., Voyant, C., Muselli, M., & Nivet, M.-L. (2010). Forecasting of preprocessed daily solar radiation time series using neural networks. In Solar Energy (Vol. 84, Issue 12, pp. 2146–2160). Elsevier BV. https://doi.org/10.1016/j.solener.2010.08.011
6. Inman, R. H., Pedro, H. T. C., & Coimbra, C. F. M. (2013). Solar forecasting methods for renewable energy integration. In Progress in Energy and Combustion Science (Vol. 39, Issue 6, pp. 535–576). Elsevier BV. https://doi.org/10.1016/j.pecs.2013.06.002