Mark Maguire

Prof. Rich Nguyen

CS 4980

Independent Research

Using Machine Learning to Predict Daily Solar Output

For my Capstone, I took CS 4980 in combination with a global sustainability class. In the global sustainability class, my group focused on the capabilities of battery storage technologies to make solar power a more viable electric generation technology. While researching solar generation, we found that large utility companies like Dominion are less likely to implement solar power due to the associated unpredictability. These companies deem solar an "intermittent resource." On sunny days solar produces excess power, while during cloudy days, solar generation plants fail to produce enough power. My goal is to use the pattern recognition power of machine learning to help reduce the unpredictability of solar generation.

The first step in this process was finding the relevant data. I wanted daily weather data and the resulting amount of solar power produced each day. I discovered a dataset containing 4,213 instances, from someone who had worked on a similar project. There are 21 columns in the dataset, 20 of which are the associated weather conditions; the 21st shows, in kilowatts, the amount of power generated from solar energy that day. From an initial assessment, some of the weather conditions appeared more applicable to my project than others. For example, while

temperature and angle of the sun seemed relevant to predicting solar output, wind speed did not seem to be relevant.

I first loaded this data into a pandas dataframe in Jupyter Notebook to begin my analysis of it. The number of kilowatts generated from solar on any given day had a large statistical spread. The mean number of kilowatts generated per day was 1,134, but the set had a standard deviation of 937 kilowatts. Next, I determined which factors correlated most strongly with the number of kilowatts generated per day. The day's shortwave radiation had the strongest positive correlation with the electricity produced, and the zenith of the sun had the strongest negative correlation. This finding makes intuitive sense: more solar radiation results in more solar energy available for harnessing. Conversely, a higher zenith of the sun would mean that the sun rays would reach the solar panels at a more indirect angle, thus yielding less power. Many factors had very little correlation, including wind speed, which was expected.

I split my data into three subsets. The first subset is for training the model, and has 70% of the instances. The other two subsets have 15% of the data each. One of these two groups will be used to train test models, and to train models that will later be used as inputs for an ensemble machine learning model.

I'm personally a big believer in the "no free lunch" theory of machine learning, and for that reasonI trained a great variety of machine learning models and to see which performed the best. At the end of the project, I coupled together my best performing models to perform ensemble

learning, in order to see if better results could be achieved.During the first semester I trained three different models: a linear regression model, a decision tree model, and a random forest model. In the second semester, I trained an SVM regressor model and a neural net. I slowly got more sophisticated with the models, to see if more robust models would perform at a higher level. The results from these models were averaged or combined as inputs to another, final model. This final model was trained on the predictions of the output of the previous models (ensemble learning).

The first model that I trained was a linear regression model. This is the simplest of the machine learning models, yet quite powerful. The linear regression model was trained on X_train and y_train (label) sets. It was evaluated on the testing set. I calculated the root mean square error, and found it to be 508 kilowatts. There was a large spread in the data, so an RMSE of 508 kilowatts is decent, but still a large amount of variance for a utility to deal with.

The second model I decided to use was a decision tree. The decision tree performed worse on the data than the linear regression model. It had a root mean square error of 554 kilowatts. Decision trees notoriously overfit datasets, and for that reason I used a random forest model for the last training model.

The random forest is like the decision tree but more advanced. Random forests are controlled by the number of features and estimators present. When training the random forest, I used random combinations of the hyperparameters for number of features and estimators. There was a large

range of results for the different random forest model's root mean square error: the best being 420 kilowatts, and the worst being 540 kilowatts. Analysing the results from the random forest provided interesting insights. The random forest that had the most number of trees also had the lowest error. This is likely due to having more trees helped to prevent one tree from overfitting the dataset.

The next model I trained was the neural net. I trained the neural net using the Python library of Tensorflow, a project developed by Google that fairly easily allows computer scientists to train machine learning models. Neural nets are presently what are considered state of the art for machine learning. The "thinking" of the machine learning model is meant to emulate the human brain. I attempted a variety of different factors when experimenting with and programming the neural net. Neural nets have hundreds of different factors, and the best way to train one is considered a "dark art" in computer science. Neural nets have a number of levels, and each level can have any number of nodes. All of the inputs go into the first level of the neural net, and they are fed through the system until they produce an output.

Every column of data you have represents an input. For my first neural net model, I had 5 layers. The last layer needs to have only one node in it, because there is only one output. All preceding layers can have as many nodes as desired. When programming the first model, I took a quasi fibonacci approach. My number of nodes went, from first layer to last, 17, 13, 8, 5, and 1. An activation function is the function used to determine the output of each node. I used a RELU activation function, which is common in neural nets. I trained the neural net with a batch size of

three, and then I trained it for 100 epochs. The results were not terrific. After the last epoch, the model had a MSE (mean square error) of almost 2.2 million kilowatts, and an RMSE of almost 1,500 kilowatts. This means it was performing significantly worse than the linear regressor from the first model.

For the second model, I took a different approach to training it. I decided to drop irrelevant columns first off. If the absolute value of the correlation between the column and the outputted solar for the day was less than .15, I decided it was unnecessary as an input. This meant the number of inputs for the neural net dropped from 20 to eight. I also had fewer layers in this model. The second model only had four layers, and each layer, from top to bottom, had 8, 6, 3, then 1 nodes respectively. I kept RELU as the activation function. I trained this model with a much larger batch size, and also with a much larger epoch. The batch size was 200, and I trained for 500 epochs. The result of this training was much better. At the conclusion, the second model had an MSE of 401 thousand (not to be confused with your retirement profile, a 401k). The RMSE of the model was 633 kilowatts, which was much closer to the models of before.

I decided to train a support vector regressor after that. Support Vector Machines are more typically used for classification problems, but can also be adapted for regression as well. Support Vector Regressors have a variety of options for kernels. For the first one, I decided to use a Radial Basis Function (RBF) kernel. There are three hyperparameters that control SVRs, and in a nested loop I iterated through almost 400 different combinations to see what SVR would yield the best results. The most successful SVR was relatively unimpressive, and had an RMSE error

of 832 kilowatts. After this, I began to train a SVR with a linear kernel, but the results were significantly worse, and the model would take minutes to train each iteration.

Finally, I combined these results into an ensemble machine learning model. I used the results from the linear regression, the tensorflow model, and the SVR model together to create three inputs to another machine learning model. Ensemble machine learning is similar to neural nets, and the different machine learning models in the ensemble system can be seen as nodes in the neural net. The results of the ensemble model were disappointing, and not even as good as some of the input models results. The RMSE was over 1000 kilowatts.

The best RMSE from any model came from the random forest, with an RMSE of 420 kilowatts. This model could be quite useful for a utility company when predicting how much power would be produced on a given day. This is equivalent to the power demand of 14 houses, so in a system the size of the entire electric grid, quite accurate. I think this does well to show that the model that will work the best does not need to be the most complex or sophisticated. Different models have different strengths. For really complicated things, neural nets can be necessary to compute the most likely outcomes, but for straightforward problems like predicting solar output, simpler models train faster and can be more accurate. Occam's Razor holds in computer science just like most other fields and parts of life.