Predicting Solar Power Generation Using the NSRDB

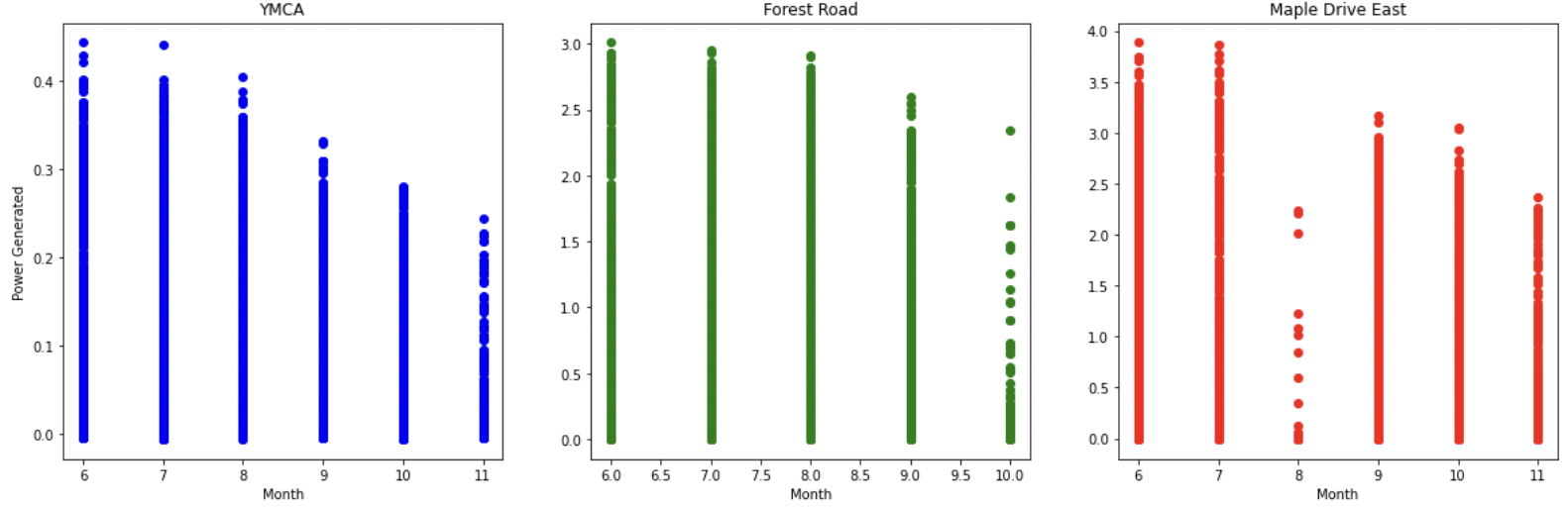
Edward Seymour, Springboard, 4 January 2023

As climate change continues to affect our lives more and more, energy companies are seeking cleaner energy sources to power the lives of their customers. The fastest growing market for clean energy is solar, as it has fewer startup costs than wind and does not garner the negative publicity of nuclear. Property owners, not just energy companies, are also keen to get into the solar market. A solar installation on the top of a home can reduce energy bills and, in places where energy can be sold back to the grid, can even pay for themselves after some amount of time. The problem comes from the unpredictability of the sun’s availability at any particular location. Not only that, but any number of factors can affect a solar cell's ability to generate electricity; humidity, wind speed, temperature, just to name a few. The variability of solar energy generation can be a huge problem for energy companies looking to invest in solar farms, since, without the proper tools, they will be unable to accurately assess how much of their demand they will be able to supply, and, as a result, whether it is a smart financial decision or not. Property owners face the same dilemma. They are unable to judge whether a large investment in solar will ever pay off at their current location.

I began this project with the plan to use data from the UK on weather and solar power generation, as well as weather data from the U.S., to create a model capable of calculating the return on investment for residential and business customers' solar projects within an error of 10% per year.

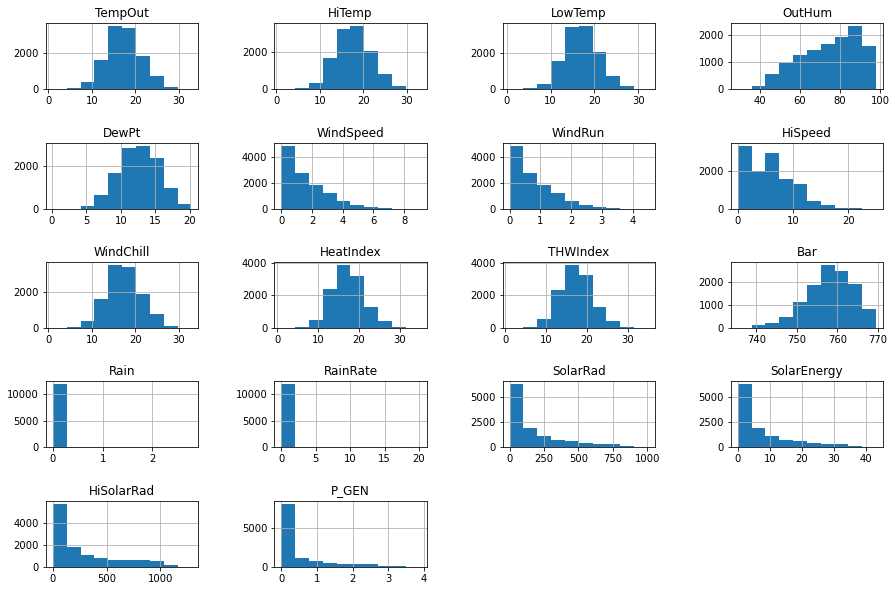
I began by cleaning the UK data of null values, as well as joining the weather and power data into a single dataset. I dropped the wind direction and wind run columns, as they had many missing values. I was uncomfortable removing any outliers at this point, as I was unsure of whether they were valid measurements or not.

Moving on, I looked at the amount of power generated at the different sites. As you can see below, The temperature is highest in the summer and lower in the winter. YMCA was a much smaller site, so it had much lower power generation than the other sites. Forest Road and Maple Drive East had much more similar amounts of generation. I dropped rows

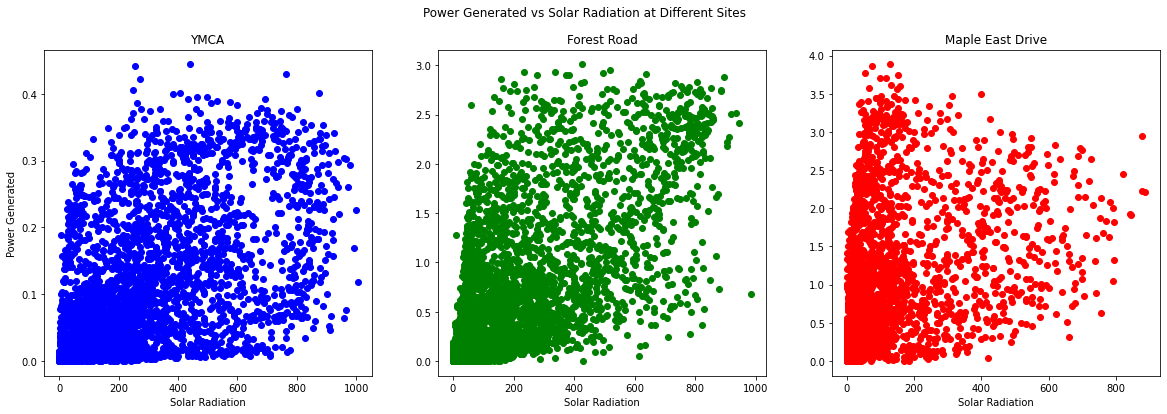


that had P\_GEN values lower than 0. I believe these were erroneous measurements.

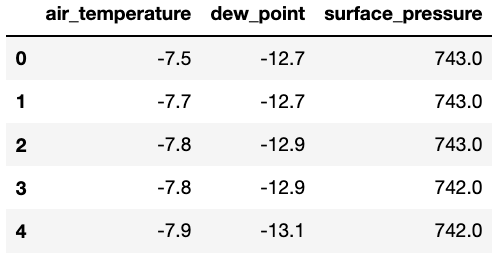
Next, I looked at the distributions of the features in the dataset. Many features had a normal distribution, while some had exponential distributions with a left or right skew. The distributions look favorable for modeling as I have a good range of scenarios for the model to consider.



Finally, I plotted the relationship between power generated and a number of the available features. Power generated versus solar radiation is shown below.

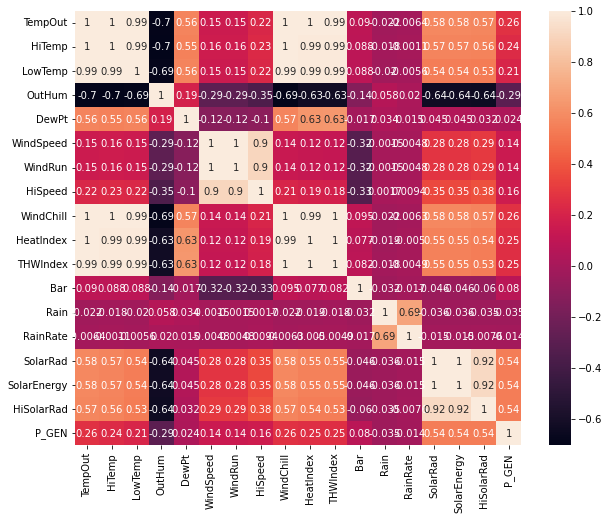


I took a detour here to work on the NSRDB data. The NSRDB data is retrieved using the H5PY package. From the NSRDB I am able to get a list of available coordinates. I can then use a cKDTree to find the closest available coordinates to any location that I want in the U.S. I then use these coordinates to retrieve selected weather columns from the NSRDB. A sample dataframe is shown below, with air temperature, dew point, and surface pressure available. The rows of the dataframe are observations made at thirty minute intervals starting at midnight, January 1st of the requested year and the columns are the requested features.



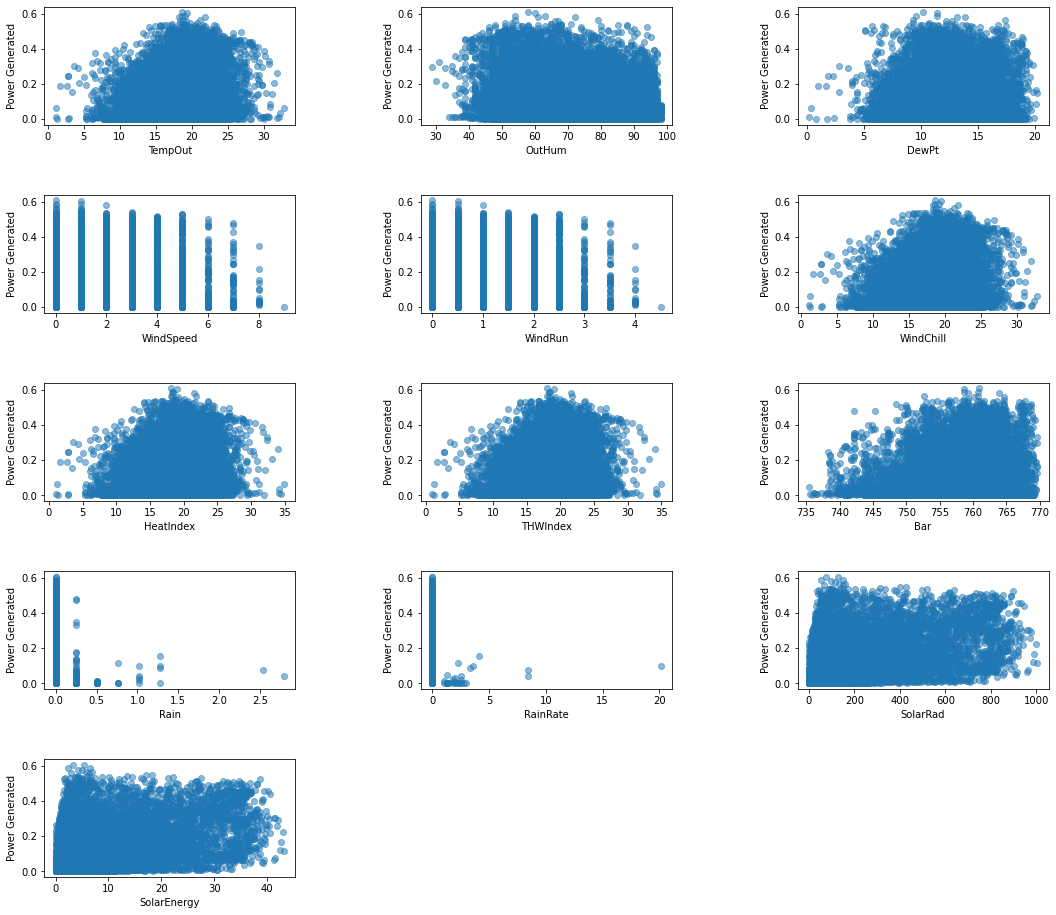
At this stage I also collected the average electricity price in all 50 states and D.C. This will be used later to calculate the amount of money that solar will save at the provided location.

Moving forward, I looked at the correlation matrix of the available features. Solar radiation and solar energy are the most positively correlated to power generated, obviously. We also see a negative correlation to P\_GEN with humidity. The index values (Heat index, wind chill, thw index) also have decent correlation to P\_GEN. The features are generally quite correlated with one another, which is to be expected with weather factors.

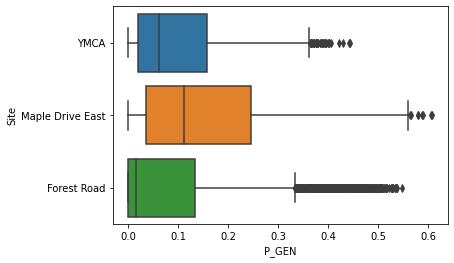


It was at this point that I decided to drop the features that were recording the extreme values of other features: HiTemp, LowTemp, HiSolarRad, etc. I left the features WindChill and HeatIndex, even though they are just functions of the other variables, as I wanted to see whether the model would find them useful.

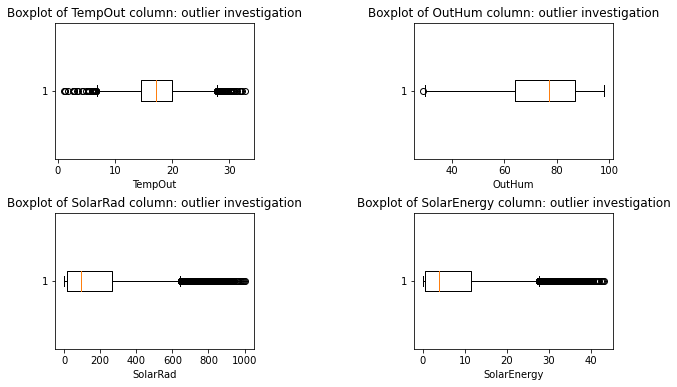
I then looked at the relationship between all of the features and the target feature P\_GEN. Those graphs can be seen below. They are quite messy and don’t provide an obvious relationship, however some graphs show a clear drop off in power generated when the explanatory feature is too high or too low.



I ended the data analysis stage by looking at the power generated at the different sites in boxplots. We can see that all sites have fairly low means and medians. They also all have outliers on the upper end of power generated. It was at this point that I began to worry slightly about the quality of the data and whether the model produced would be successful.



I decided to look closer at the outliers in the P\_GEN column. I needed to know whether they were valid measurements or not. To do this, I looked at the other columns. If the other columns had similar outlier occurrences, then that gives some credibility to the idea that those times were particularly good for solar power and not that those instances are times when the power measurement sensors went awry. Those distributions are shown below



As you can see, particularly in solar radiation, the distribution is almost identical to the P\_GEN distribution. It is indeed possible that some days were much sunnier than others. The measurement period was from June to November, so it is understandable that the earlier days would be excellent for solar power while most of the remaining months would be poor.

A little more confident in the data, I split the data into “x” and “y” components, with P\_GEN being my “y” and the remaining features my “x”. I then split those variables into training and testing sets with a 75/25 split, respectively.

I started the modeling by using a dummy regressor to get some baseline metrics to judge my future models against. I used a median method, instead of a mean, since I had a lot of outliers in the P\_GEN column. The median can account for the outliers better than the mean. With that, I got a mean absolute error of 0.091 and an r2 score of -0.155.

The four models that I decided to test were linear regression, support vector, regression, gradient boosting regression, and random forest regression. I am using a regression model because my target variable is a continuous variable, so classification would not be appropriate. In each case, I set up a pipeline to do a randomized grid search to find the best parameters. I used a randomized grid search instead of a full grid search because I had many parameters that I needed to test. A full grid search would have taken far too long. I will be testing whether to use a StandardScaler or a MinMaxScaler and how many PCA components to use in all pipelines.

The linear regression model produced a mean absolute error 0.075 on the test set, which is better than our baseline. This is a good start. The best metrics for this model were eleven PCA components and a MinMaxScaler. The r2 value for this model was 0.333 on the test set.

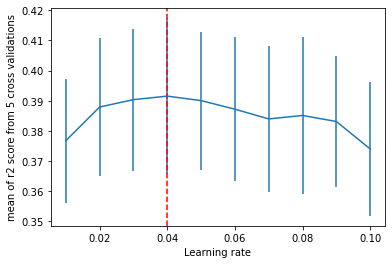
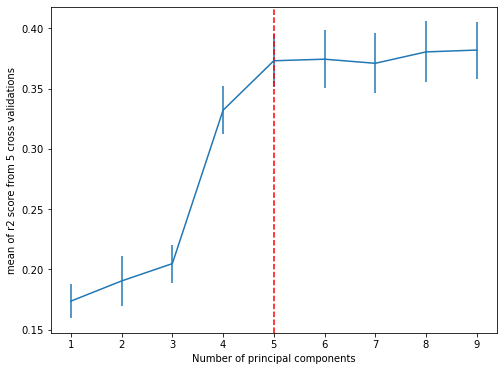
The support vector regression model produced a mean absolute error 0.0.089 and an r2 score of 0.285 on the test set. This is still better than the baseline, but worse than the linear regression model. The best metrics for this model were 5 PCA components, a MinMaxScaler, the ‘rbf’ kernel, an epsilon of 0.1, a degree of 4, and a c value of 0.3.

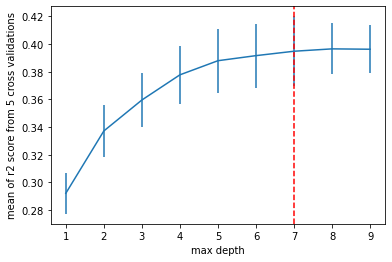
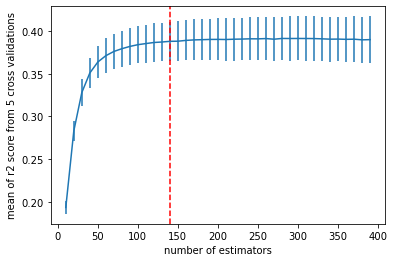
The gradient boosting regression model produced a mean absolute error 0.074 on the test setThe best metrics for this model were nine PCA components, a MinMaxScaler, 150 estimators, a max depth of 2, and a learning rate of 0.1. . The r2 value for this model was 0.357 on the test set.

The random forest regression model produced a mean absolute error 0.068 on the test set. The r2 value for this model was 0.399 on the test set, so this model is performing the best of all four on the test set. The scores for the training set are also much better than the other four. The best metrics for this model were thirteen PCA components, a MinMaxScaler, a max depth of nine, and 80 estimators.

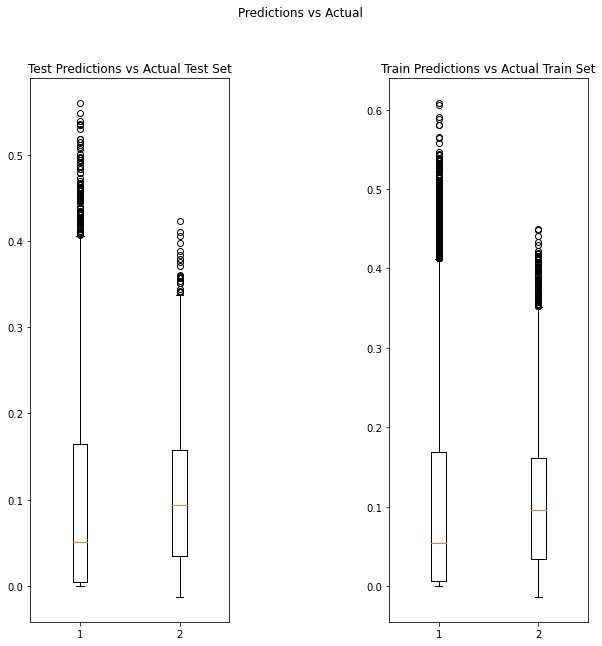
I ran these random searches many times over and it appeared that random forest and gradient boosting were neck and neck on being the best. Sometimes the random forest would perform the best, sometimes it was the gradient boosting regressor. Going forward, I am going to use the gradient boosting regressor.

Now that I know gradient boosting regression is the model that we should go with, I narrowed in on the best values for each parameter. I decided to do cross validation on each individual parameter to find the best value one at a time. Those graphs are shown below with lines at the chosen values to use for the final model. The parameters were 5 principal components, a learning rate of 0.04, a max depth of 7, 140 estimators, and a minmaxscaler

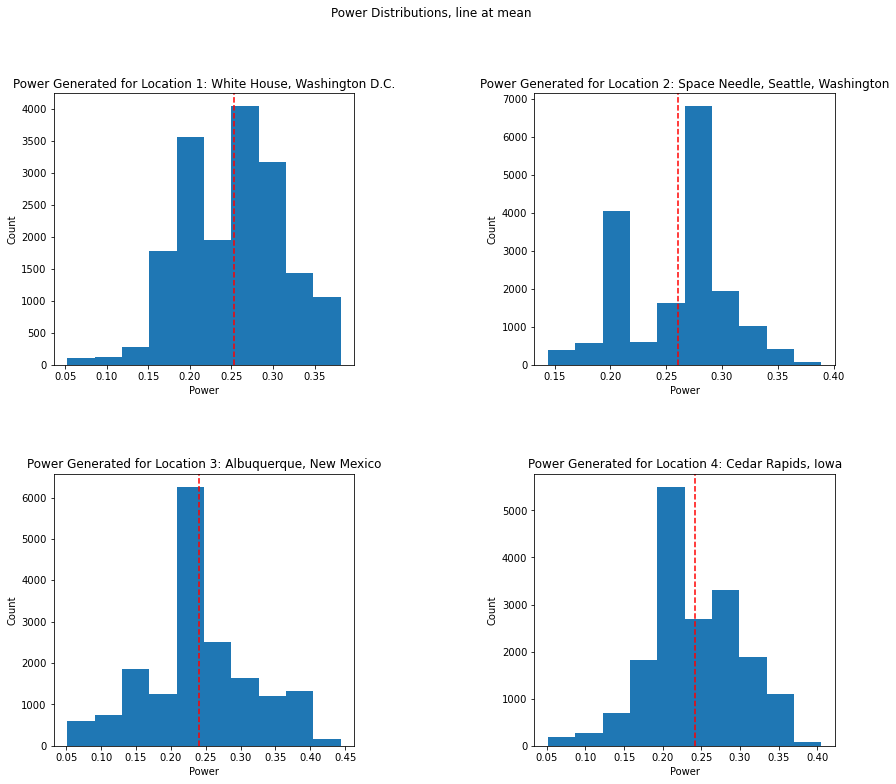




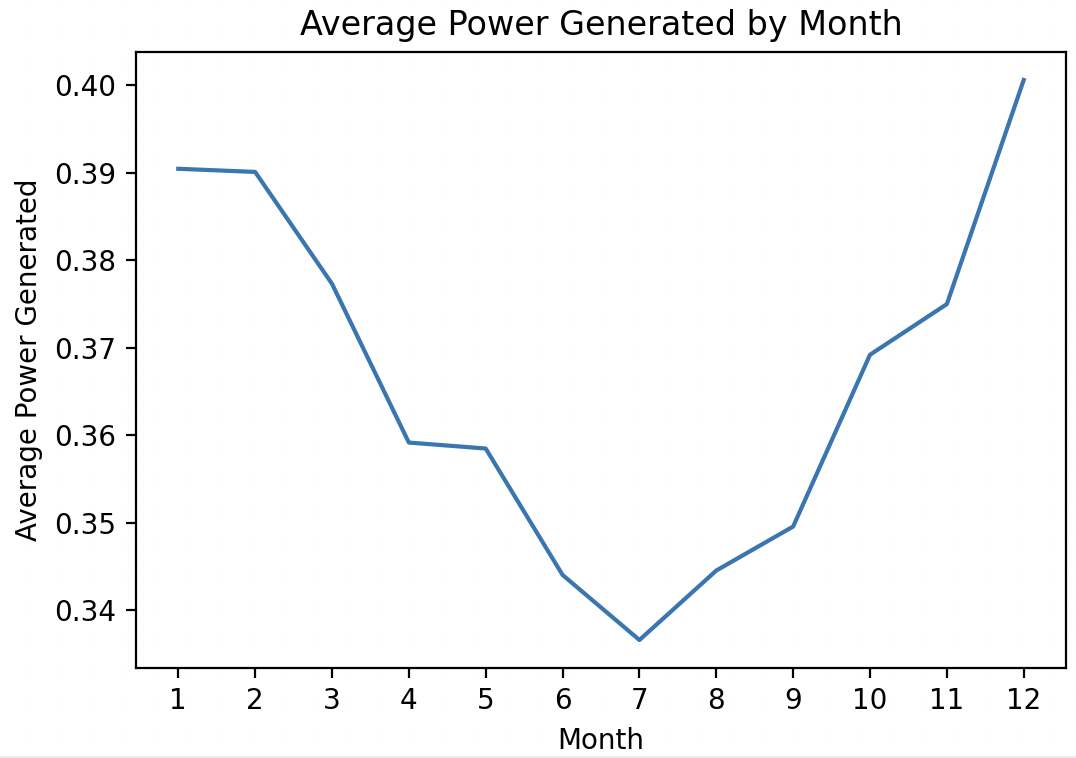
After finding those values, I retrained the model and used it to predict the values for the test set. The boxplots for the test and train sets are shown below; predictions are shown on the left and the actual values are shown on the right. As you can see, the distributions are quite similar. The r2 score was now 0.409, so the parameters that I found here were better than the ones found by the randomized search.



I decided at this point to try removing the outliers from the P\_GEN column. I had hoped that the model would perform better after doing this. In fact, it improved the model quite noticeably. Using the same model parameters, removing the outliers caused the r2 score to increase to 0.441, a great improvement.

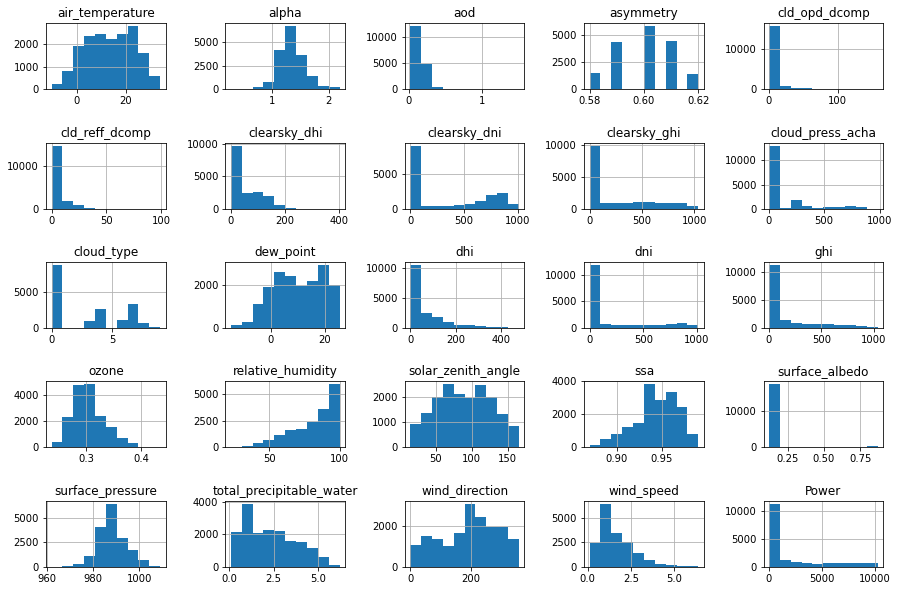
I was now ready to use this model to predict solar power using weather observations from the NSRDB. The goal is to be able to return the amount of money that would be saved at a given location with a solar installation of a certain size. I could then use the cost of installation provided by the user to tell them how long it would take to pay off their investment. Below is shown four graphs for different locations in the U.S.

The time to pay off these installations averaged around ten years, which is consistent with the information that I could find online[1]. At this point, I felt pretty confident in my model. I wanted to see one last thing, though. I wanted to see the amount of power generated over the year. This is when I saw the flaw. Below is the graph of the average amount of power generated each month at a location in Washington D.C. As you can see, the summer months should be the best for power generation, yet they seem to be the worst. January and December are the best, completely defying logic! Something was wrong.

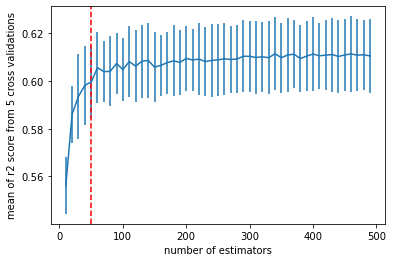
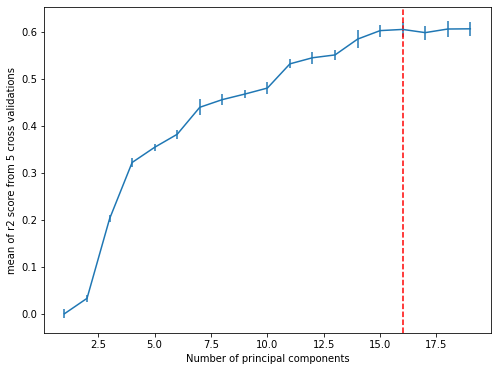


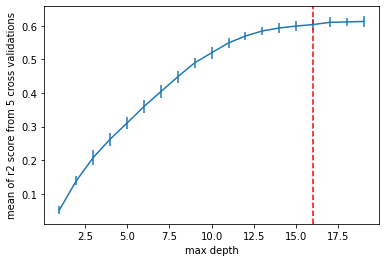
I went back to look at the data and find what was causing this issue. I was unable to find the smoking gun and had to resign to believing that the data itself was just not well suited to explaining my target variable well enough to be able to be extrapolated in the way that I needed it to be.

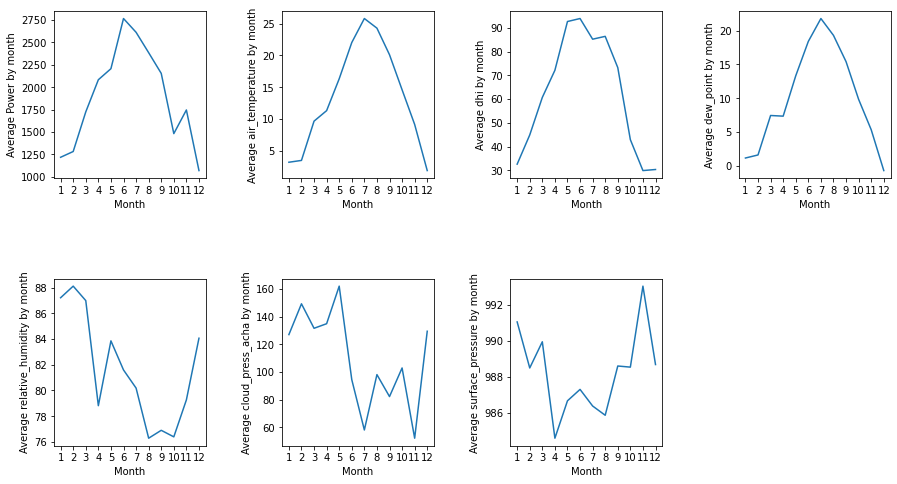
My solution: get new data. I was lucky to quickly find new data, in the U.S., no less. The data, from the E.W. Brown Solar Facility in Kentucky. The data from this source is just the power generation data. Although, it’s in the U.S. and I know its location. So I can use the NSRDB to get the weather data! Below is a number of graphs showing the distributions of the weather variables for the E.W. Brown Solar Facility. These distributions are not too dissimilar from the UK data. Although, I now have a lot more features to explain my target variable.



I decided to re-do my randomized searches to see if the best model was still what I thought. Actually, this time, a random forest regressor came out on top every time! So, gradient boosting was out and random forest was in. I did the cross validation on each parameter again and found that the best parameters were 16 principal components, a minmaxscaler, 50 estimators, and a max depth of 16.



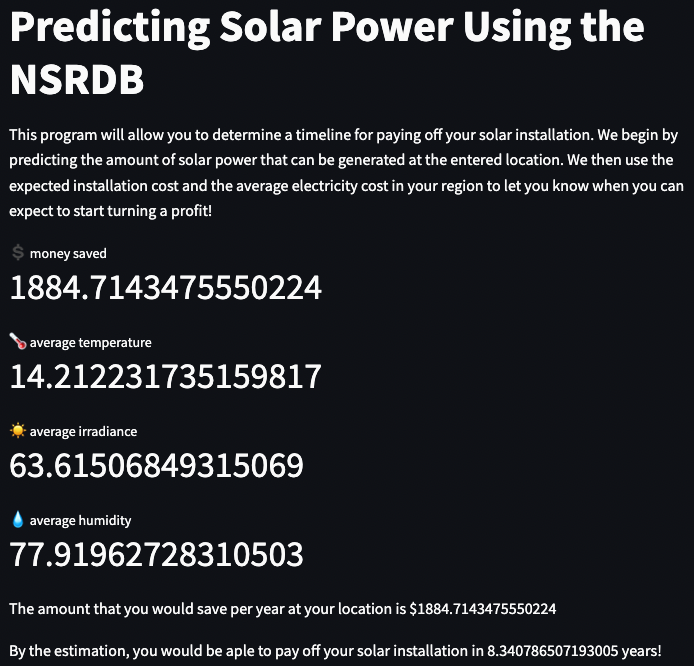


Checking the power generated each month, once again, the model is now correctly identifying the summer months as peak power generating months and the winter as the drop off. This is exactly what I would expect. The time to pay off the investment is still averaging around ten months, which is awesome! The final r2 score was 0.656, which is quite unbelievable. This is so much better than what was obtained with the previous training data.

The application that I built for this model can be reached at:

<https://eddy12321-solar-roi-apppredicting-power-using-the-nsrdb-7bnxjc.streamlit.app/>

In this application, the user is prompted to enter their address, the size of their installation in kW, and the cost of their installation. With that information, I use an API called Geoapify to turn that address into coordinates. I then find the closest available coordinates in the NSRDB and get the weather for the location. I use the model and the E.W. Brown training data to predict the amount of power that can be generated at that location. I then tell the user how much money they can expect to save each year and how long it will take for them to pay off their investment. A screenshot of the app output is shown on the next page.



Please give the app a try!

This project started out with the idea to use UK data to predict US data. This did not provide the intended results. The time to pay off the investment came out sensibly, however this is most likely due to the manner in which this is calculated. The time to pay off the investment is only concerned with the total amount of power generated in the year, not whether you’re generating that power in the winter or the summer. After replacing the UK data with data from the E.W. Brown Solar Facility, our model performed exactly as expected. The model that performed the best out of the four that we tried was random forest regression. The best parameters were; max depth of 13, 50 estimators, MinMaxScaler, and 16 principal components. The Streamlit app that I built works well and returns the information that I set out to return at the beginning of this project. In the future, to make the model better, I would like to add more weather features and provide more features describing the type of solar panels, their direction to the horizon, and whether they are self-cleaning or not. I am confident in my final product.

Citations

1. <https://www.solarreviews.com/blog/how-to-calculate-your-solar-payback-period#:~:text=Average%20solar%20panel%20payback%20period%20for%20homes%20in%20the%20US,the%20state%20they%20live%20in>., SolarReviews, 4 January, 2023
2. <https://data.london.gov.uk/dataset/photovoltaic--pv--solar-panel-energy-generation-data>
3. <https://data.openei.org/s3_viewer?bucket=nrel-pds-nsrdb>
4. <https://lge-ku.com/our-company/community/neighbor-neighbor/ew-brown-generating-station>