Solar Panel Capstone

Optimizing cost efficiency for residential installations

By Zachary Brown

Abstract

Solar panel installations are one of the most effective ways for an individual to lower their carbon footprint, however, they often cost over $10,000 which is unaffordable for many. My goal in this project was to use solar panel installation data from the Lawrence Berkeley National Laboratory's Tracking the Sun program to identify what aspects of a solar panel installation can be controlled to maximize cost efficiency for a person living in Austin Texas. After compiling the data, expanding the features, and comparing a wide range of regression models, an XGBoost regression model was developed that identified the following guidance:

- Design the solar panel installation with a high inverter loading ratio
- Identify and secure any rebate or grant that may be available
- Consider buying a solar panel model with lower conversion efficiency
- Purchase the largest configuration of solar panels possible for the house
- Consider scheduling the installation for July or December

The model had root mean squared error (RMSE) of 35 million which was due primarily to three major outliers that appear to be typos in the total installation cost. To expand on this work in the future I think it would be valuable to design a tool that takes into account weather, location, this historical installation data, and average electricity use to make a recommendation on how large someone should build their solar panel array, what rebates or grants may be available, and how long it will take for the solar panels to pay for themselves.

Introduction

As global temperatures rise, many people are looking for ways they can affordably reduce their carbon footprint to help slow climate change. One effective change homeowners can make is to install solar panels for electricity generation, however, the average cost for a residential solar panel installation is around $16,000 according to Forbes.com. For many homeowners the only way to invest in solar panels is to ensure that they are as cost-effective as possible so that they can potentially make their money back over the life of the solar panels.

The Lawrence Berkeley National Laboratory maintains a program called Tracking the Sun which collects nationwide solar panel installation data. In addition to creating a yearly report with their insights, they also host the data for public use. This dataset includes features such as the total installation price, the system size in direct current (DC) kilowatts (KW), any rebates or grants

obtained, the solar panel tilt angle, rotation angle, inverter size, inverter loading ratio, installer, installation state, city, and zip code, electric utility territory, etc.

For some background on how a solar panel system works, first the photovoltaic module (solar panels) absorb sunlight and convert it into DC electricity. Next an inverter converts the DC electricity to alternating current (AC) which is transmitted to the house first, and the grid second if there is excess. Solar panels can be mounted either on the ground, or more typically for residential installations, on the roof. They are angled to maximize the amount of sunlight they can capture, factoring in the tilt angle (facing straight up vs facing sideways) and the azimuth angle (360° rotation along the north, south, east, west axis).

My goal in this project was first to take the most recent residential installation data from this Tracking the Sun dataset to calculate the cost per KW for each installation which would be my metric for cost efficiency. I would then develop regression models to help identify what customers in Austin Texas can do to minimize that metric for their own solar panel installation.

Methodology

Loading Data

I began this project by importing and joining the parquet datafiles hosted by the Tracking the Sun program. I looped through the files for each state, saved the parquet file, loaded them into individual dataframes, added a column identifying the state for each, and then joined them into one unified dataframe. I then reduced the data to only the relevant residential installations. After doing this I dropped the customer segment column since there was no information to glean from that feature anymore. The data was then split by installation year, and since there were only 2100 entries from the year 2021, I chose to use data from 2020 and 2021 so that I'd have ~240,000 entries to work with. A column for installation month was then created to capture any seasonality associated with pricing.

Creating the key metric

Since the metric of this project is price per KW, any field missing that value was dropped. The dataset encodes missing values as -1, so I replaced those with 0 so that they didn't impact the final price per KW metric. I then limited the dataset to only rows where the system size was greater than zero, because A) it doesn't make sense that someone would install solar panels that generate zero or negative electricity and B) that will be the denominator in the equation calculating price per KW. At this point I created the key metric column: price_per_kw using the following equation:

$$price\ per\ KW\ = \frac{total\ installed\ price - rebate\ or\ grant}{system\ size\ DC}$$

Once this column was created I dropped the total installed price column from the data because it is a key driver of price per KW that is essentially out of the control of the customer.

Tuning location features

My goal was to maximize cost efficiency for someone in Austin Texas, so I wanted to make sure that my models could account for location. I started by creating dummy columns for each state present in the dataset. Digging deeper into location, I trimmed 9 digit zip codes to only the first 5 digits, and then I created dummy columns for each unique zip code with more than 30 entries. The zip code column was then dropped to avoid collinearity. Finally I checked the individual cities within Texas and found that there were no entries for Austin, so I decided to drop the city column entirely and use zip code as my smallest location feature.

Feature engineering

Feature engineering consisted primarily of five steps: dummying categorical variables, train-test splitting the data, imputing missing values, scaling features, and then feature selection (Figure 1).

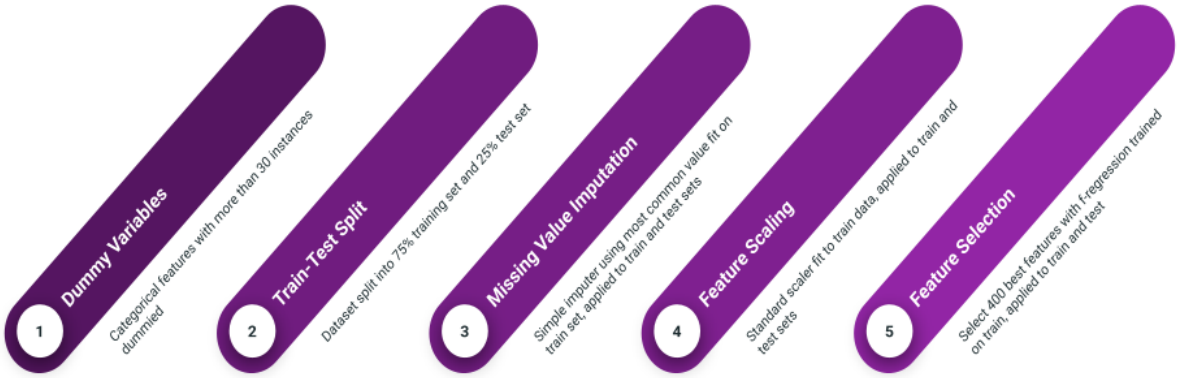

Figure 1. The steps used during feature engineering are described in order.

Before creating dummy variables I checked the proportion of each feature that was missing and I dropped any features missing 30% or more of the data. I counted the number of unique values in object type categorical columns and created dummy columns from any column with fewer than 25 unique values. For any remaining features I created dummy columns from any unique value with more than 30 occurrences and then dropped the original column to avoid collinearity.

Next I performed a 75%/25% train-test split on the data so that I could evaluate my final model appropriately. I then used scikit-learn's simple imputer to impute the most common value for each training set column, then imputed both the train and test set using this value. Next I fit a standard scaler on the train set and then transformed the train and test sets. Finally, I narrowed the features down from almost 2900 to the 400 most important using f regression. The x_train, x_test, y_train, and y_test sets were then saved separately to avoid data leakage throughout model development.

Exploratory Data Analysis

To begin my exploration of the data I began with a simple box plot of the cost efficiency to get a feel for the distribution of the data (Figure 2).

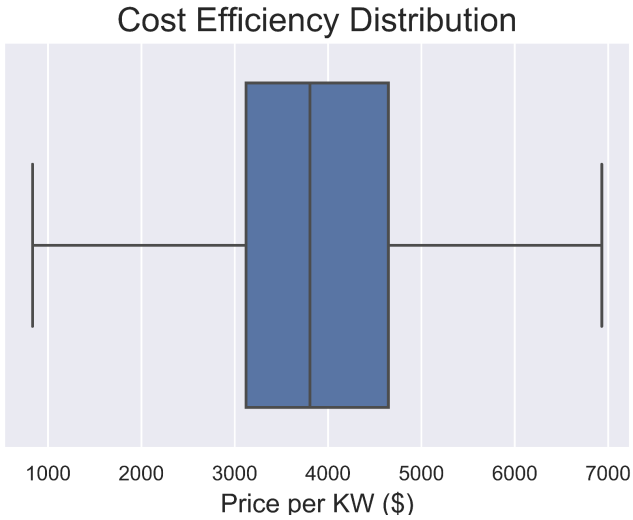

Figure 2. The distribution of price per KW for this dataset, excluding outliers.

Next, to get an idea of whether Texans should expect to pay more or less per KW compared to the rest of the US, I plotted empirical cumulative distribution functions (ECDFs) for both (Figure 3). I then verified with a Shapiro-Wilk test that both portions of the data were normal and confirmed with an independent t-test that the two are significantly (t statistic = -8.33, p-value = 8e-17).

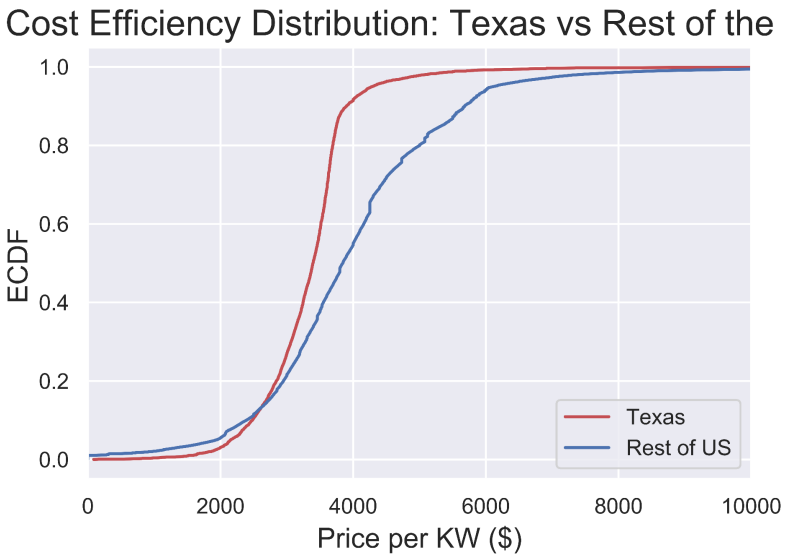

Figure 3. ECDFs for Texas installations compared to non-Texas installations showing that Texas has a more narrow distribution at a lower average.

Next I prepared boxplots of installations by month (Figure 4) and then verified via chi-squared test that there is a significant correlation between the installation month and price per KW (test statistic: 8730078, p-value: 0.0).

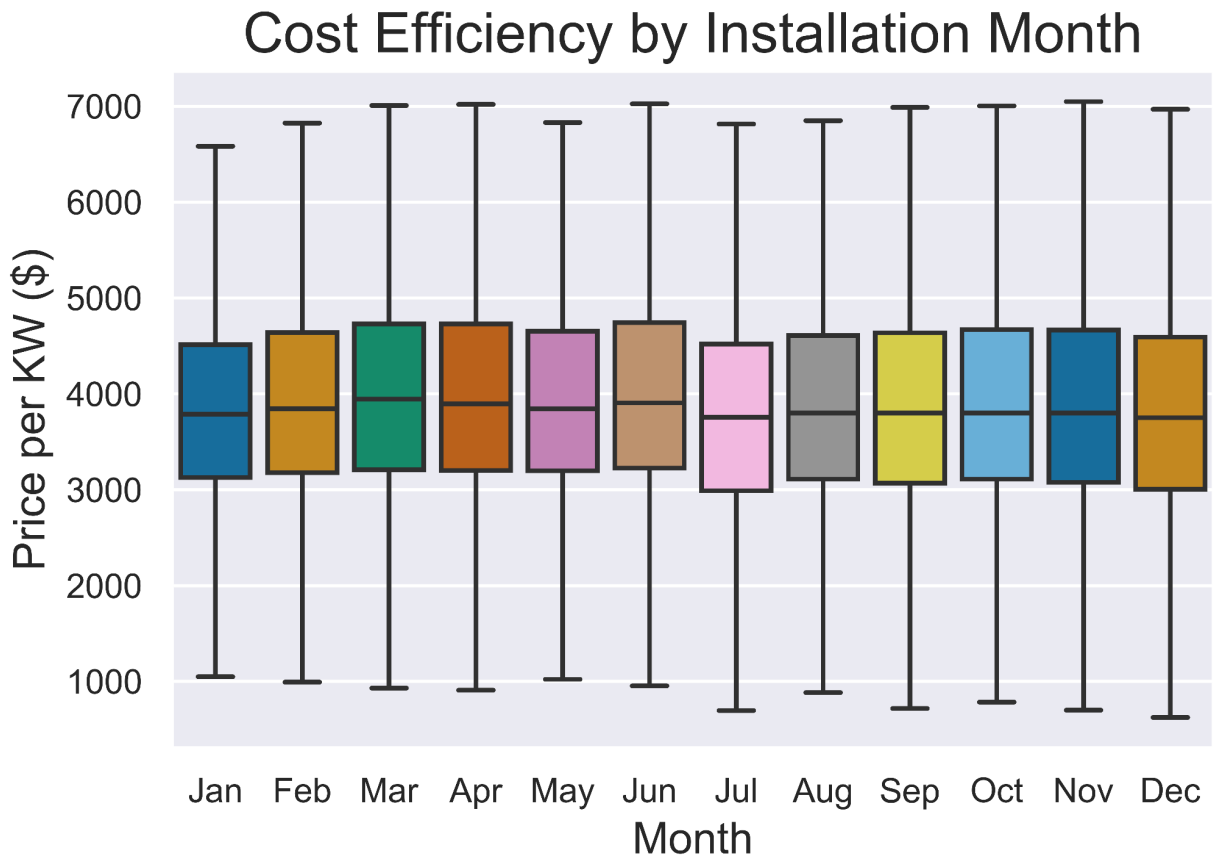

Figure 4. The distributions of cost efficiency vs installation month are plotted with outliers hidden.

I proceeded to loop through any categorical variables that allowed yes/no/missing responses and prepared boxplots of each followed by t-tests to identify whether any correlation was significant. These variables included whether the installation was an expansion of an existing system or not, whether the system was multiple phase, whether the system included tracking equipment, if the system was ground or roof mounted, whether the system was owned by a third party, if it was self installed, whether there were additional modules installed, if the modules were building integrated, if the panels were double-sided, if there were additional inverters installed, if the inverter was a micro-inverter, a hybrid inverter, or was built in to the photovoltaic system, and finally, whether there iwa a DC optimizer built in. Of these features, self installation, additional modules, building integration, additional inverters, and hybrid inverters all had significant correlations with cost efficiency.

Next I looped through the continuous features and prepared scatterplots to visualize any obvious correlations with cost efficiency. Two that showed obvious negative correlations were the number of inverters (Figure 5) and the system size (Figure 6).

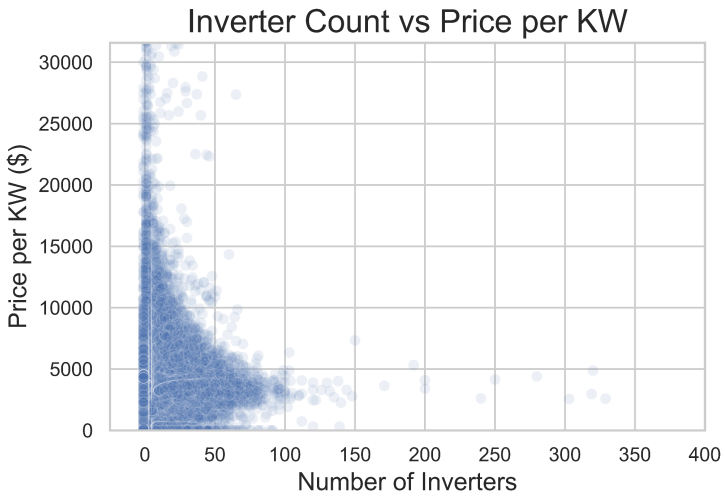

Figure 5. Inverter size is plotted against cost efficiency, showing a clear negative correlation.

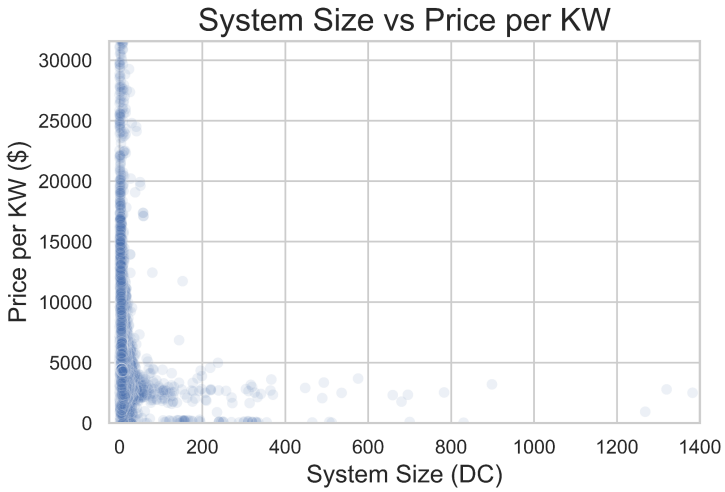

Figure 6. System size is plotted against cost efficiency, showing a clear negative correlation.

Other continuous features were less obvious, so ordinary least squares (OLS) regression lines were fitted and the p-values for the intercept and slope were used to determine whether the correlation was significant. In all cases where I performed this analysis the R^2 was less than 0.00, so while it was clear that none of these features singlehandedly predicted cost-efficiency, they may still have some significant correlation. Module efficiency was the first feature I tested this way; Table 1 shows the resulting parameters and Figure 7 shows the plot and regression.

Table 1. Slope, intercept, and each corresponding p-value are shown for the OLS regression between module efficiency and cost efficiency.

	Statistic	**p-value**
Slope	3880.995467	2.791065e-06
Intercept	3254.944534	5.904827e-87

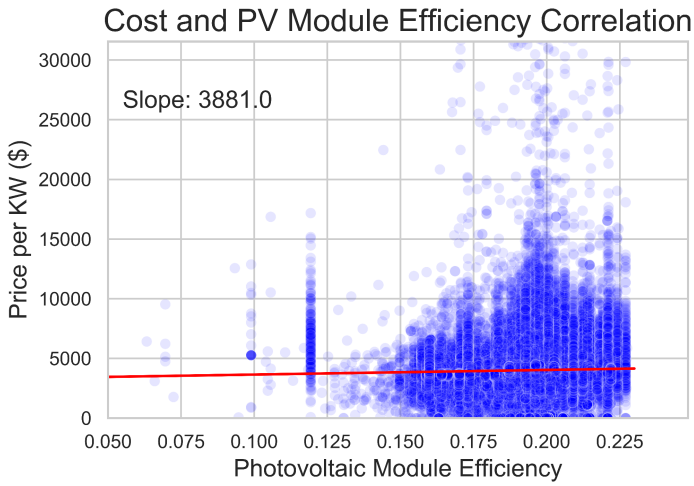

Figure 7. The correlation between module efficiency and cost efficiency is shown with the OLS regression in red.

Advertised capacity was similar, showing a slight positive correlation with cost efficiency/ Inverter loading ratio was a particularly surprising feature. When plotted against cost efficiency (Figure 8), there is no clear slope to the data. However, the OLS analysis shows a significant negative correlation as described in Table 2.

Table 2. Slope, intercept, and each corresponding p-value are shown for the OLS regression between inverter loading ratio and cost efficiency.

	Statistic	**p-value**
Slope	-517.405285	2.207527e-37
Intercept	4607.379181	0.000000e+00

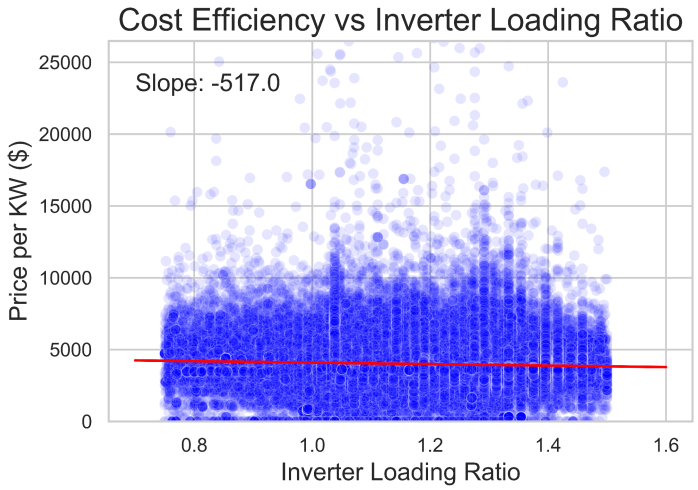

Figure 8. The correlation between inverter loading ratio and cost efficiency is shown with the OLS regression in red.

This concluded my preliminary analysis of the data prior to modeling. These results proved to be quite insightful when analyzing the final model results.

Modeling

My plan for this project was to screen a wide range of regression models and narrow down to the best model for this dataset. As a first step I loaded just the x_train and y_train datasets and then split them into 10% train and 90% test subsets. I used random search CV with 60 iterations to tune hyperparameters for each model of interest. Figure 9 is a graphical visualization of the resulting RMSE values for visual comparison. Table 3 shows the models tested, the hyperparameters that were tuned, and the RMSE values plotted in Figure 9.

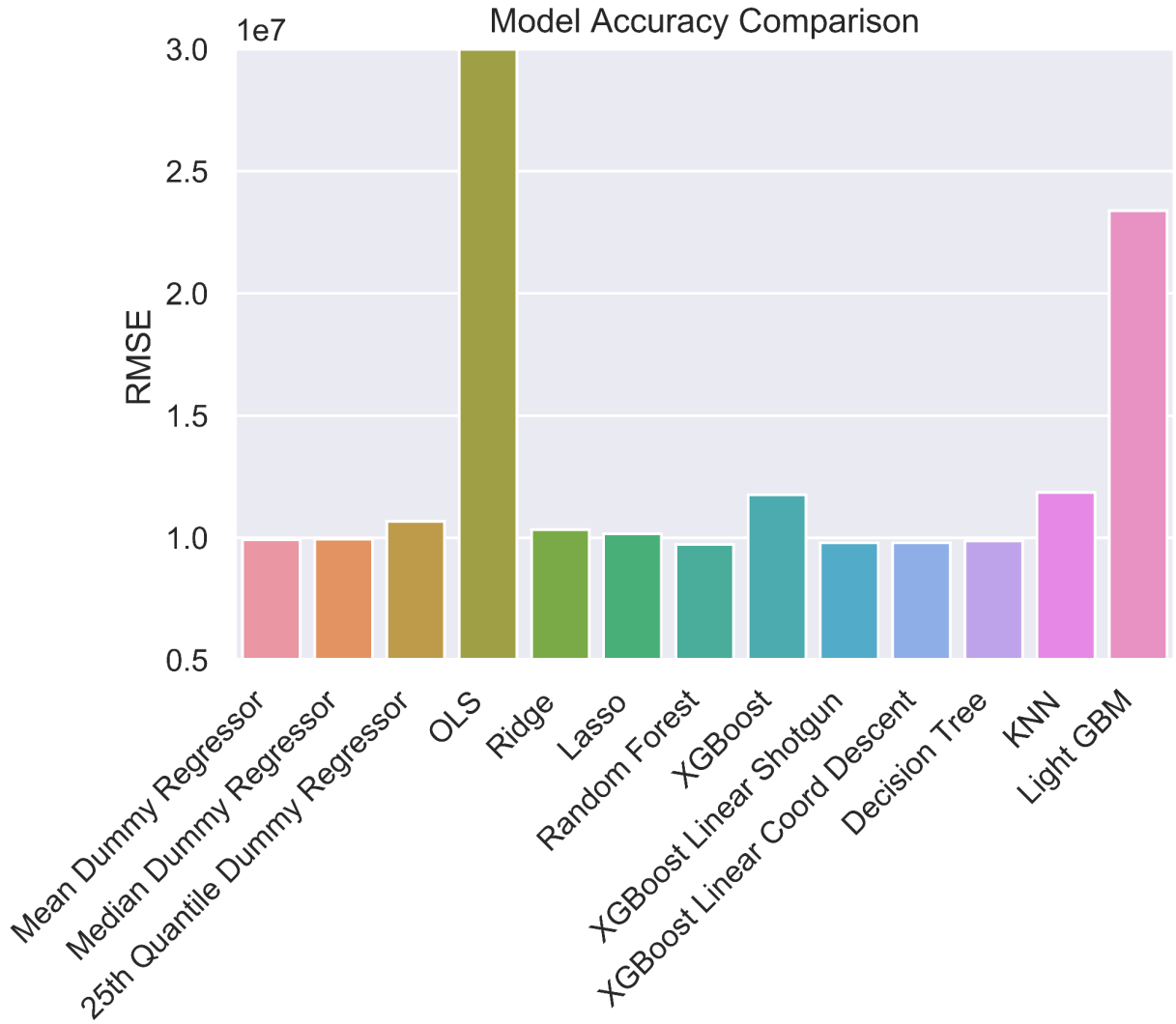

Figure 9. RMSE is plotted for each model using the best hyperparameters identified through randomized search CV. Models were trained on only 10% of the available data.

Table 3. The models initially trained on 10% of the data are shown with the hyperparameters that were tuned and the best resulting RMSE when tested against the remaining 90%.

Model	Hyperparameters	RMSE
Mean Dummy Regressor	N/A	9,923,409
Median Dummy Regressor	N/A	9,941,871
25th Quantile Dummy Regressor	N/A	10,670,605
Ordinary Least Squares	N/A	1.2650560e+19
Ridge Regression	alpha	10,327,496
Lasso Regression	alpha	10,155,465
Random Forest Regression	max_features, max_depth, min_samples_leaf, n_estimators	9,720,312
XGBoost Regressor	n_estimators, max_depth, eta, colsample_bytree	11,759,799
Linear XGBoost Regressor - shotgun updater	reg_lambda, reg_alpha, feature_selector	9,788,061
Linear XGBoost Regressor - coordinate descent updater	reg_lambda, reg_alpha, feature_selector	9,789,561
Decision Tree Regressor	max_depth, min_samples_leaf	9,865,568
K Nearest Neighbors Regressor	n_neighbors	11,856,353
Light GBM Regressor	num_leaves, n_estimators, max_depth, learning_rate	23,385,497

After comparing these initial results most of the models were retrained using the same hyperparameters, this time on 80% of the data. They were tested against the remaining 20% and the resulting RMSE for each is compared in Figure 10.

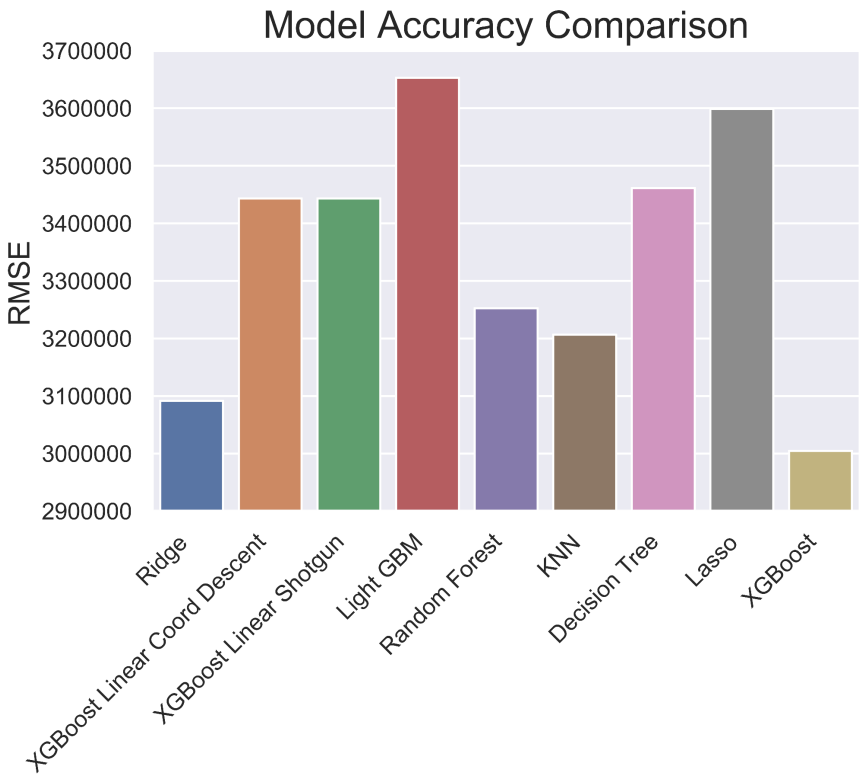

Figure 10. RMSE is plotted for various models using hyperparameter settings from the initial tuning and trained on 80% of the available data.

Now, having identified the best options as ridge, KNN, and XGBoost regressor (I included Light GBM because in earlier iterations of this notebook it performed much better than this), I explored the hyperparameter tuning results to determine whether any should be retrained on more data. Figure 11 shows the hyperparameter tuning results for XGBoost and Figure 12 shows the same for Light GBM.

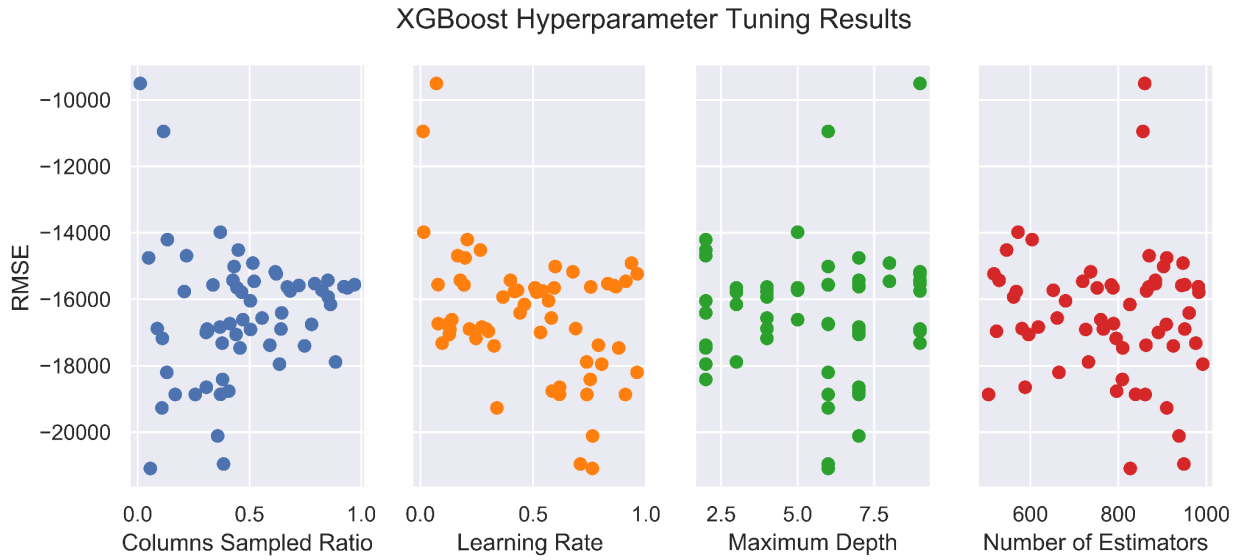

Figure 11. Hyperparameter tuning values for XGBoost are plotted against the resulting RMSE values.

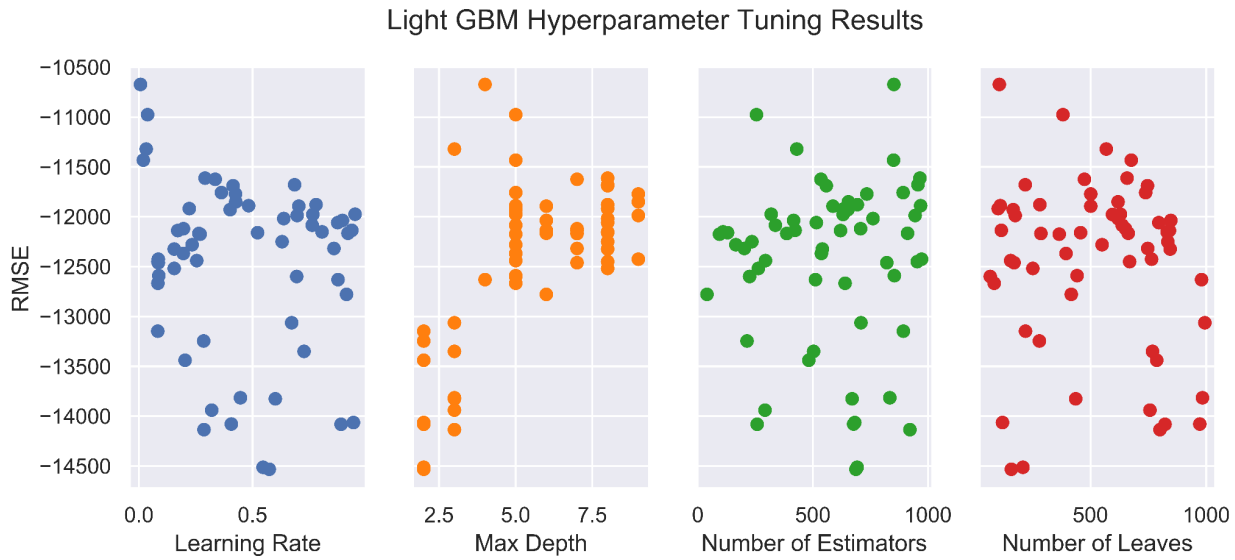

Figure 12. Hyperparameter tuning values for Light GBM are plotted against the resulting RMSE values.

These results suggested that I probably found a sweet spot for max_depth and n_estimators in the XGBoost model since there are no obvious trends, so I decided to lock those where they were. I decided to lock the num_leaves and n_estimators in Light GBM for the same reason. The remaining hyperparameters in those four models were tuned again, this time fitting 80% of the data to try to squeeze out any improvement possible. The resulting model test scores are plotted in Figure 13.

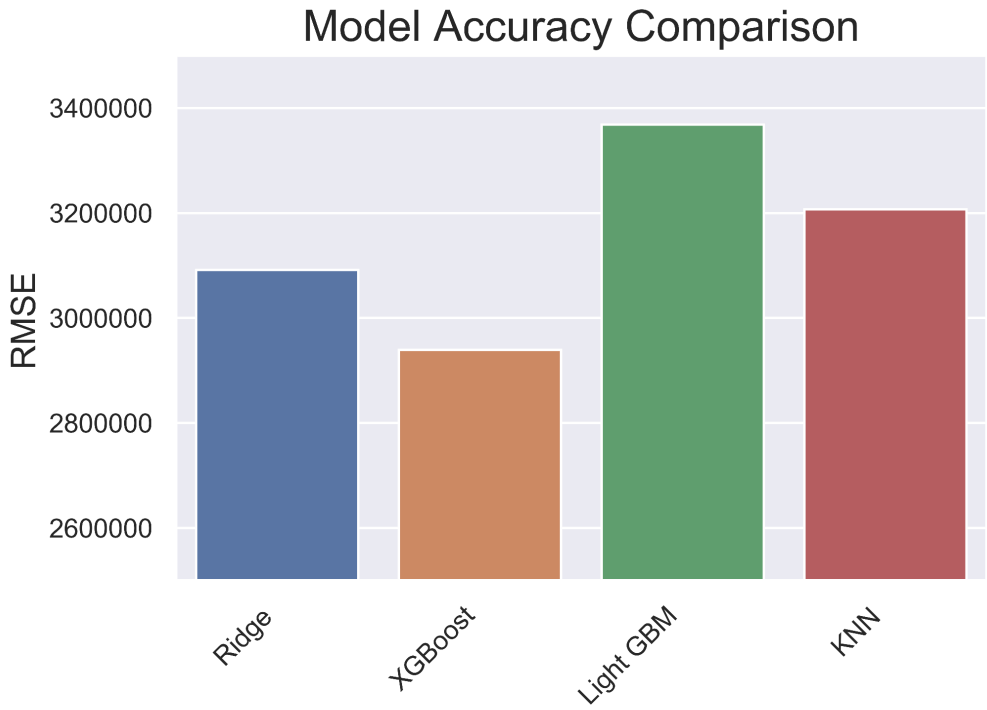

Figure 13. The four remaining models are plotted showing their RMSE after hyperparameter tuning on 80% of the data.

Insights and Performance

After hyperparameter tuning was complete the x and y test datasets were loaded into the notebook. The final XGBoost regressor model was trained on the full training set and tested on the test data. The RMSE increased from 2.96 million within the training set to 35.05 million when exposed to the separately saved test data. I suspected that there might have been outliers triggering this increase, so I plotted the predicted y values against the test values and identified 3 outliers (Figure 14).

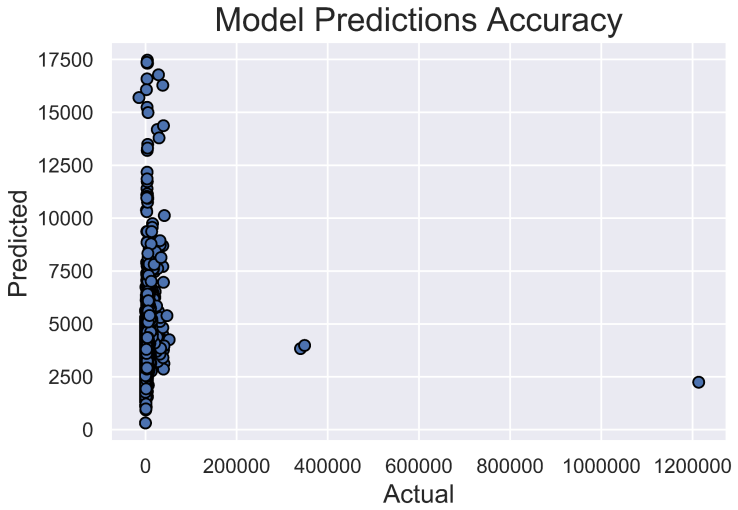

Figure 14. The y-test values are plotted against the predicted values, showing three very significant outliers.

When I identified the outliers and examined their entries in the original dataset I found that all three listed total installed prices in the millions, whereas most entries were in the tens of thousands with an average of around $25,000. Given that those three outliers listed typical values for system size but the prices were 3427200, 2631400, and 8255000, it seems likely that they were entered with two extra zeros at the end of their true prices. Nonetheless, I proceeded to plot the feature importances for this model to gauge what features it found most important to predicting cost efficiency (Figure 15).

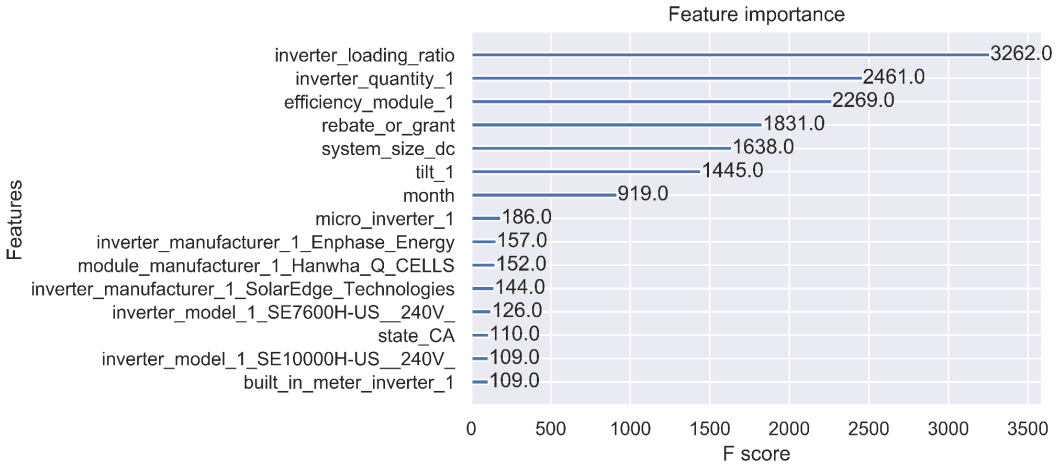

Figure 15. The top 15 features of my XGBoost model are listed by F score.

As shown in the exploratory data analysis, all of the top seven features were identified as having significant correlations with price per KW. Of those top seven features, inverter loading ratio, inverter quantity, rebate or grant, and system size all had negative correlations with price per KW (improving cost efficiency). Module efficiency was the one feature that had a positive correlation with price per KW (decreasing cost efficiency). Installation month and tilt are more complex.

If we dig deeper into those features with negative correlations, it becomes apparent that there is a volume discount for equipment, where the price per panel/inverter decreases as the number of panels/inverters increases. That discount is directly reflected in inverter quantity and system size. Inverter loading ratio is related but slightly more complicated. The inverter loading ratio reflects the maximum amount of DC current the solar panels can create vs the maximum amount of DC electricity the inverters can convert. Say a solar panel installation has an inverter loading ratio of 1, so at peak hours the solar panels can create 5 KW of DC electricity and the inverter can convert that full 5 KW to AC. That's great during peak hours, however, during off-peak hours (most of the day) this system will only capture a fraction of this capacity. In a system with the same inverter capacity but an inverter loading ratio of 1.3, some of that peak hour DC will be lost because the inverters can't handle it, but during the rest of the day that system will produce 30% more energy than the first system. This is shown gaphically in Figure 16.

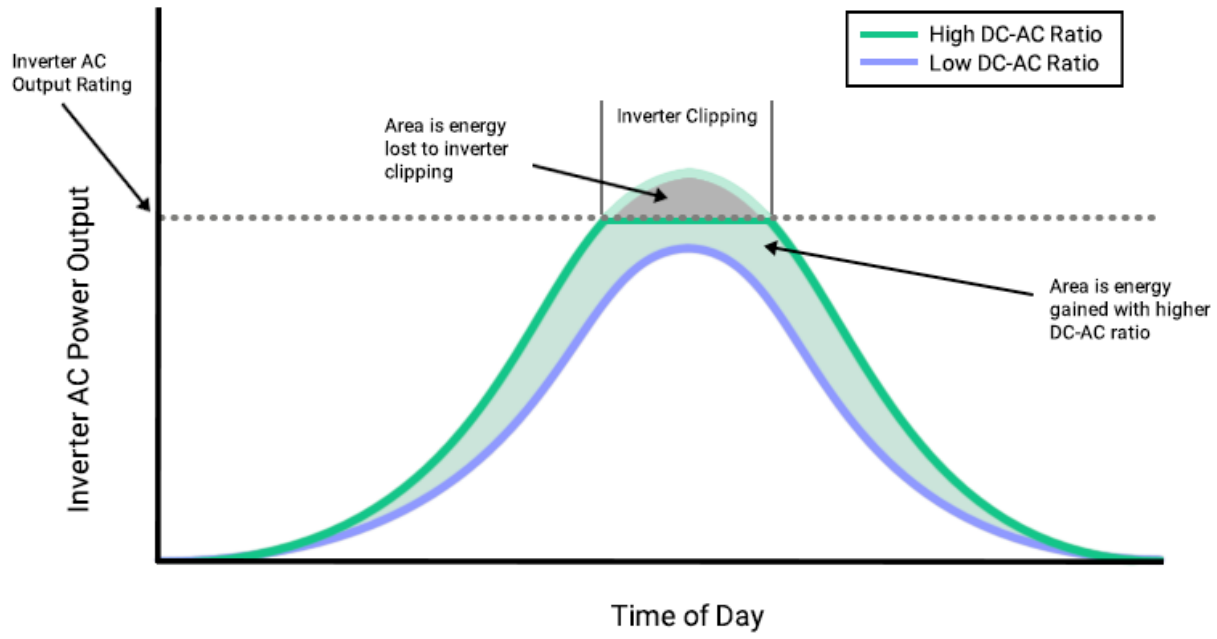

Figure 16. This plot shows the advantage of having an inverter loading ratio greater than 1. **Source**: https://aurorasolar.com/blog/choosing-the-right-size-inverter-for-your-solar-design-a-primer-on-inverter-clipping/

Module efficiency has a positive correlation meaning that as module efficiency increases, the cost per KW does as well. While buying less efficient panels reduces cost efficiency for the

installation, it likely increases the overall cost efficiency for the life of the solar panel array. This is one feature the customer will have to judge based on their unique situation because it could have a significant impact on their long-term savings with more efficient solar panels.

To understand installation month it helps to revisit Figure 8 which shows that there are some months with lower average price per KW than others. July and December stand out as the most inexpensive months of the year, but the entire second half of the year trends lower than the first half. One possible explanation for this could be that holiday specials may help drive down cost around the 4th of July and during the December holiday season without being reflected as rebates or grants.

Tilt is much less clear than any other feature. As shown in Figure 17 there are specific angles that are much more frequent than others.

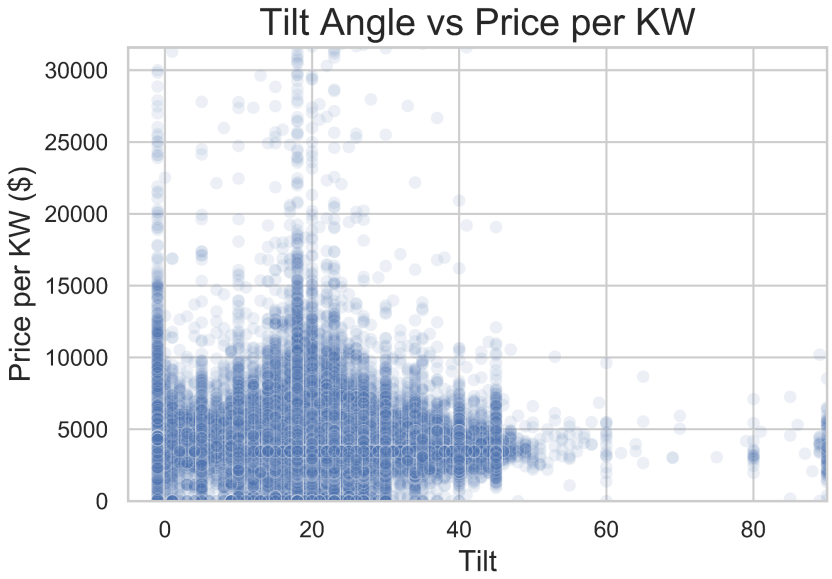

Figure 17. Price per KW is plotted against tilt angle.

It seems most likely that although there is a correlation between tilt angle and cost efficiency, it may be a feature of how many data points we have at each angle rather than a true direct correlation.

Conclusions

This study has identified multiple features of a solar panel installation that can help a customer maximize the solar electricity generated for their money. The most important factor is the overall size of the solar panel array that is chosen. The customer will want to install as large an array as possible, while taking into account their budget and the limits of any money they can recoup by selling excess electricity back to the grid. Along with the array size, they will want to choose a relatively high inverter loading ratio to help minimize inverter cost while still converting a large amount of captured sunlight. In terms of the up-front installation costs, they will want to take

advantage of any rebates or grants available to them and they should consider whether the added cost of higher efficiency panels will pay for themselves over the long term, or if it makes more sense to stick with cheaper, lower efficiency panels. Finally, if the customer is not on a tight schedule, they should target July or December for their installation, as those months seem to generally offer better cost efficiency.

In the future I would like to take this work a step further and develop a recommendation tool to help make solar panel installations even more accessible for homeowners. Right now there are some tools available to help potential customers predict the output they could get from a solar panel array. However, I'd like to take it even further by taking the customers budget, average monthly electrical use, location, and electrical provider to recommend the size of their solar panel array, the number of inverters they should install, any rebates or grants they are eligible for, and provide the length of time it would take for the solar panels to pay for themselves. There are a lot of factors to consider when deciding whether to install solar panels or not, and I think a tool like this would make that decision much less daunting.