Predicting Solar Energy Potential with Machine Learning

Image: Unsplash

Author: Samuel Miller
Mentor: Michael Darling
Data Science / Machine Learning
CodingNomads

Table of Contents

(Clickable Links)

Abstract

With the effects of climate change continuing to worsen, solar energy represents one of the most viable energy alternatives to fossil fuel resources. Locating optimal sites for new solar farm construction remains imperative to the viability of the grid-decarbonization process.

Direct Normal Irradiance (DNI) remains a primary feature for predicting the solar output potential in a given region. DNI measures the amount of solar radiation received by a surface that is held perpendicular (or normal) to the rays that come in a straight line from the sun.[1] Unfortunately, DNI data can be challenging to obtain and computationally expensive to work with. Building on the work of Abiodun Olaoye[2] and Pasion et al.,[3] this project implements machine learning (ML) techniques to construct models for predicting solar output.

In particular, this paper demonstrates the effective use of Recursive Feature Elimination with Cross-Validation[4] in optimizing the feature selection process. Working with data collected at 12 sites over 14 months[5], the best model overall, Random Forest Regressor,yielded an R-squared score of .668.

Introduction

The US Department of Energy projects the demand for electricity will grow approximately 30% by 2035[6]. Keeping up with this growth will require massive investments in renewable energy resources if we do so in a sustainable fashion.

Solar energy presents a viable option to actualize this endeavor. According to the same report, it can potentially produce 40% of the country's electricity supply by 2035 – up from the 3% it currently supplies.

To further develop solar energy infrastructure, new sites must be located with the highest potential to reliably produce solar electricity. Predictive modeling, based on various features, can help determine such sites.

---

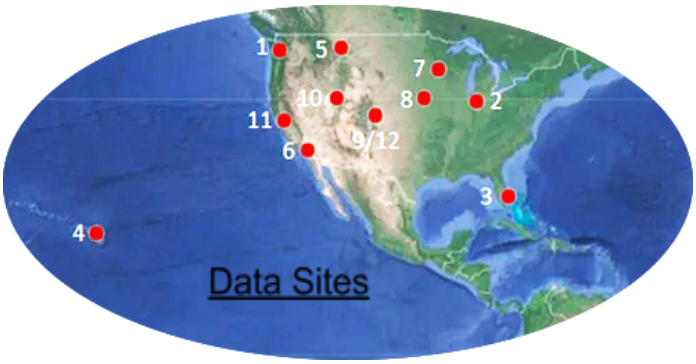
[1] *The Role of the Direct Normal Irradiance (DNI) Forecasting in the operation of Solar Concentrating Plants*. Cogliani, E. Energy Procedia Volume 49, 2014, Pages 1612-1621.
[2] (Olaoye) Olaoye,Abiodun. "Predicting Solar Power Output Using Machine Learning Techniques". Towards Data Science. 2022.
[3] (Pasion et al.) Pasion, C.; Wagner, T.; Koschnick, C.; Schuldt, S.; Williams, J. & Hallinan, K. Machine Learning Modeling of Horizontal Photovoltaics Using Weather and Location Data. Energies 2020.
[4] (RFECV) Recursive Feature Elimination with Cross Validation.
[5] Data acquired from Kaggle. 2022.
Source: Pasion, C.; Wagner, T.; Koschnick, C.; Schuldt, S.; Williams, J. & Hallinan, K. Machine Learning *Modeling of Horizontal Photovoltaics Using Weather and Location Data*. Energies 2020, 13, 2570; doi:10.3390/en13102570.
[6] (SFS) *Solar Futures Study*. U.S. Department of Energy Solar Energy Technologies Office (SETO). 2021.

Typically, a chief indicator of solar availability is DNI. However, various factors can degrade the practical utility of this metric in predicting the viability of candidate sites. As stated by Pasion et al., "...irradiation data can be time-consuming to measure at a specific site, and prediction of irradiation can generate forecast errors, [which] may be unsuitable for accurate PV performance analysis."

In this project, I applied machine learning techniques to predict the solar potential of a given region based on factors other than DNI. Working with data collected at 12 sites over 14 months, I determined which of the 16 observed variables correlate most strongly with the dependent variable, "PolyPwr"(power output). In addition to correlation mapping, I applied Recursive Feature Elimination with Cross-validation (RFECV) to eliminate those features that benefit the modeling process the least.

I have extracted a pared-down list of nine independent variables to better model predictions through the above feature selection techniques. The selected list includes the following features: date, time, latitude, month, humidity, ambient temperature, [atmospheric] pressure, cloud ceiling, and winter (binary class).

After determining the optimal feature list, I fitted four separate predictive models to the transformed data. These models include Dummy Regressor (Dummy), Random Forest Regression (RFR), Stochastic Gradient Descent (SGD), and Multi-layer Perceptron regression (MLP).



A Note on Randomness:

Randomness is integral to data science and machine learning. You will notice throughout this paper that I have introduced it into most stages of the project. This is because randomness, along with ample iterations over the data, allows the model to analyze and learn from the data in a less patterned fashion. By random sampling many different approaches and taking the average, as does RFR, for example, the model takes advantage of the law of large numbers to reduce overfitting.

In evaluating the models' performance, this project makes use of the following metrics: absolute error (MAE), root mean squared error (RMSE), R^2 (R2). This provides for direct comparison with the previous works cited since they also use these metrics. Throughout this paper, I most often use R2, which "is the proportion of the variation in the dependent variable that is predictable from the independent variable,"[7] as it conveys an easier-to-understand reading of each model's predictive capacity.

[7] (R2) - Coefficient of determination.

RFR performed best overall, with a score summary of:

Scores for RFR Model:

```
MAE      : 2.625
RMSE     : 4.061
R2       : 0.668
Expl_Var : 0.668
```

Related Work

As mentioned previously, this project exists in conversation with the work of Olaoye. His work uses the following models: Random Forest, Light Gradient Boosting Machine, and a Deep Neural Network. He then uses a stacked ensemble, including each of those models. For a baseline model, he uses K-Nearest Neighbors.

Model	R^2	RMSE	MAE
K Nearest Neighbors (baseline)	0.618	4.403	2.971
Deep Neural Network	0.662	4.142	2.709
Random Forest	0.670	4.095	2.787
Light Gradient Boosting Machine	0.677	4.054	2.723
Stacked Model	0.681	4.024	2.670

Model performance on the unseen test data. (Olaoye)

Besides using different models, my methodology differs from that of the related works chiefly regarding the feature selection phase of the project. His work involves a significant feature engineering process to determine the optimal features for modeling. This report uses feature selection algorithms such as RFECV and an ExtraTreesRegressor (ETR).

I am grateful for the work of Abiodun Olaye and Pasion et al. on this subject. Their methodology offered significant inspiration and guidance for conducting my research.

Initial Exploratory Data Analysis

Readers can find all of the work/code for this report in the notebook.

After importing the necessary libraries and reading-in the data, I began the initial exploratory data analysis (EDA).

Key Insights:

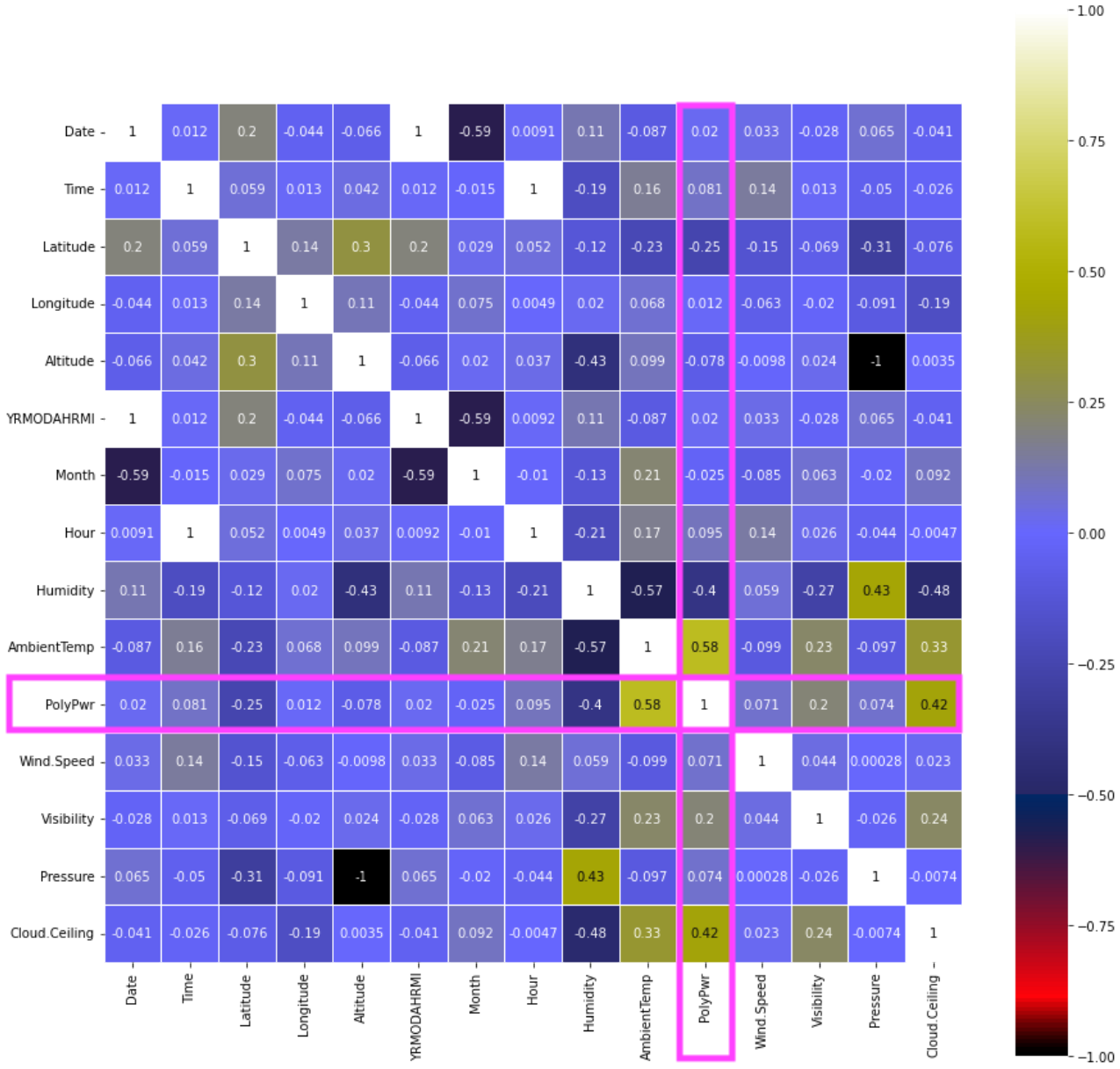
- Features: 17
- Rows: 21045
- Null Count: 0
- Dtypes: “Location” and “Season” are both objects.
- *Target Variable:* “PolyPwr”

```
RangeIndex: 21045 entries, 0 to 21044
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Location        21045 non-null  object
 1   Date            21045 non-null  int64
 2   Time            21045 non-null  int64
 3   Latitude        21045 non-null  float64
 4   Longitude       21045 non-null  float64
 5   Altitude        21045 non-null  int64
 6   YRMODAHRMI      21045 non-null  float64
 7   Month           21045 non-null  int64
 8   Hour            21045 non-null  int64
 9   Season          21045 non-null  object
 10  Humidity        21045 non-null  float64
 11  AmbientTemp     21045 non-null  float64
 12  PolyPwr         21045 non-null  float64
 13  Wind.Speed      21045 non-null  int64
 14  Visibility      21045 non-null  float64
 15  Pressure        21045 non-null  float64
 16  Cloud.Ceiling   21045 non-null  int64
dtypes: float64(8), int64(7), object(2)
memory usage: 2.7+ MB
```

The initial query shows that both “Location” and “Season” will require One Hot Encoding (OHE)[8] so we can use them for further analysis. In its own section, I elaborate more on the rationale of this encoding process below. Fortunately, there are no missing values in the data. The target variable is “PolyPwr,” representing power output by watts in 15-minute intervals. I have assigned it to the variable “y.”

[8] (OHE) Generally referred to as One-Hot Encoding, the particular tool used for this project is the Get_dummies function included in the Pandas package.

The following heat map illustrates the correlative relationship of the data. For instance, it shows that the target variable correlates positively with “AmbientTemp,” “Cloud.Ceiling,” and “Visibility.” It correlates negatively with “Humidity” and, to a lesser degree, with “Latitude.”



Pre-Processing

Train / Test Split

```
In [2]: # X, y
        y = df.loc[:,'PolyPwr']
        X = df.drop(labels = 'PolyPwr', axis=1)
        # train / test
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Before proceeding with EDA, I assigned the independent variables to “X” and gave the target variable to “y.” I then split the data into training and testing sets–80% and 20%, respectively.

Dropped Columns

I have dropped several less useful columns to mitigate the negative impact of high dimensionality. These include “Location,” “Longitude,” “Altitude,” and “YRMODAHRMI.”

The “Location” variable is highly correlated with the target variable. However, it will not generalize to unseen data at new locations, especially if a model relies too heavily on its calculations.

Unlike latitude, “Longitude” offers little predictive value to the modeling. I reached this conclusion by researching the domain and reviewing the two previous studies conducted on the data set. We can also confirm this by observing its lack of correlation to the target variable.

As one would imagine, “Altitude” negatively correlates with “Pressure.” In the context of this project, the former fails to explain the data variance in a manner distinct from the latter. Lastly, we’ll drop “YRMODAHRMI” (year, month, day, hour, minute) due to its redundancy to the variables already tracking many of its elements.

Scaling

The data requires scaling for improved correlational analysis. For this, I used Standard Scalar,[9] offered by Sci-Kit Learn. Set to default, this scaler transforms the features by “removing the mean and scaling to unit variance”(Scikit-learn). To prevent data leakage, I will include this transformer in the project pipeline rather than in total to the train/test sets.

[9] Standard Scaler. Scikit-learn.

Cross-Validator

```
cv = KFold(n_splits = 3, shuffle=True)
```

When performing cross-validation, I have used K-fold Cross-Validation[10]. This involves splitting the training data subsets, called K-folds, or “folds.” One fold is held out as a validation set, while the other splits are used for training. This process occurs once for each split, holding out a different fold for each iteration until each has been used for validation once. While the default setting of K-fold doesn’t shuffle the data before splitting it, our cross-validator will. To mitigate the effects of any phantom structures within the data, I have set the K-fold cross-validator to shuffle the data set before splitting it.

One-Hot Encoding

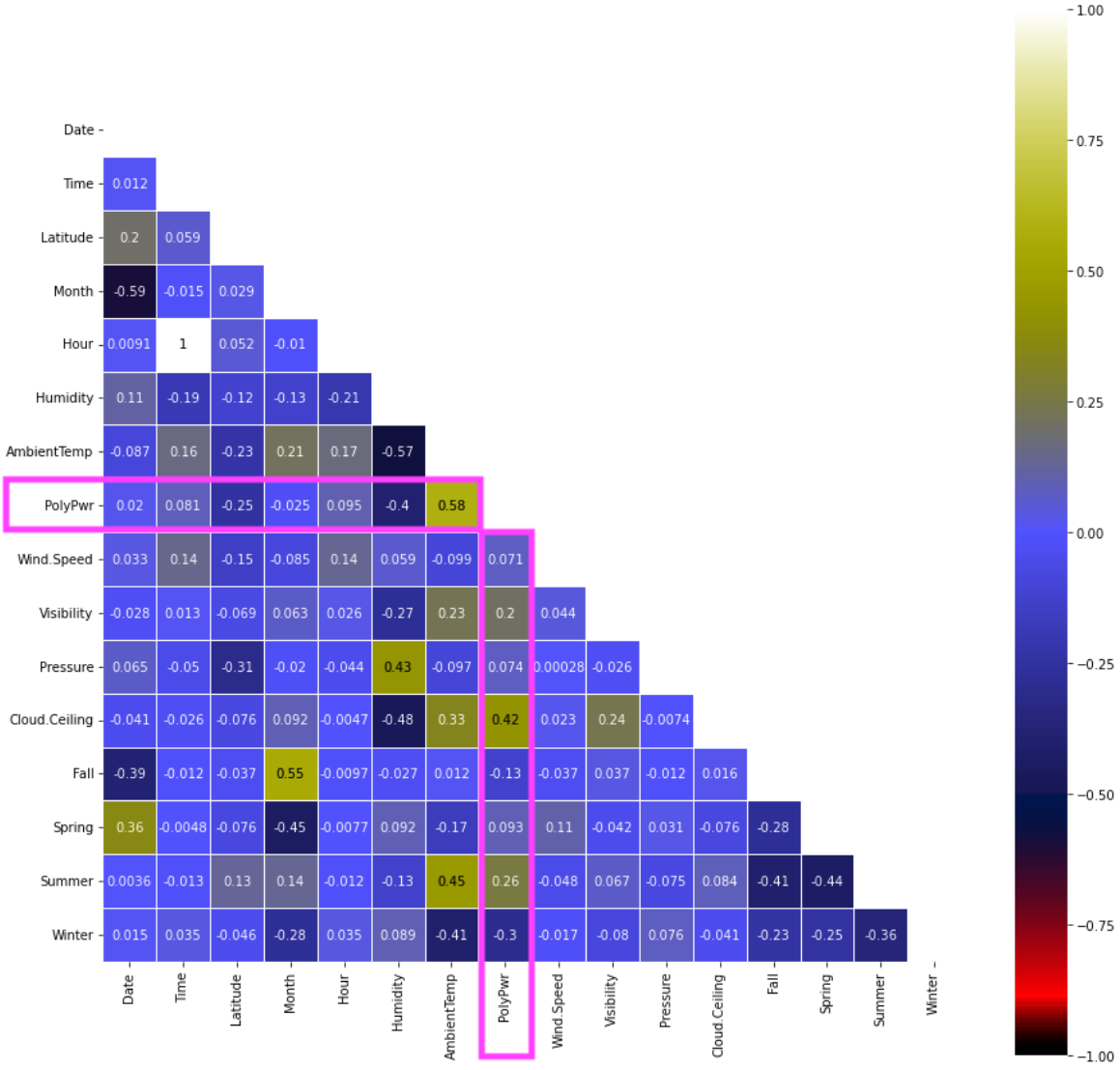
The “Season” feature contained data of the type “object." To include it in the modeling process, I encoded it numerically. For this, I used one-hot encoding (OHE)[11]. I use the “get_dummies()” function from the Pandas library, which performs OHE. Since there are four seasons, we create a new column for each season and drop the original “Season” column. For example, if a given row of data was collected in the winter, the “Winter” column will have a value of “1,” while the three other season columns will each have a value of “0.” This binary coding system allows us to relate it to our other variables mathematically.

[10] K-Fold Cross-Validation.
[11] (OHE) - One-Hot Encoding. Using the Pandas get_dummies() function.

Exploratory Data Analysis

Continued

Once I had pre-processed the data, I resumed EDA. A new heat map reflects the data without the dropped columns. Note that this map masks the mirrored data in the upper triangle:



Heat map of pre-processed data.

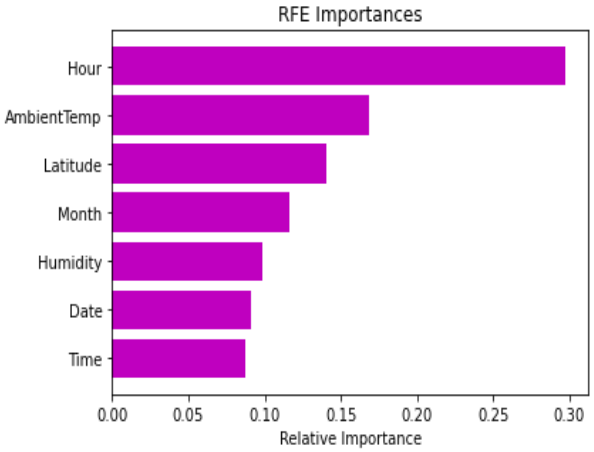
The heat map shows that the target variable correlates positively with “Summer” and negatively with “Winter,” which intuitively makes sense.

Feature Selection

Recursive Feature Elimination

In addition to heat mapping the correlative relationship of the features, Recursive Feature Elimination (RFE)[12] will provide insight into the importance of each. This algorithm acts as a wrapper function, using a model to estimate the most important features. For this project, I have chosen the ensemble, Extra-Trees Regressor (ETR)[13].

The main difference between ETR and RFR is that it introduces more randomness. Also called "Extremely Random Trees," ETR is similar to RFR in that it also picks out random input values from the training set. However, it does not use bootstrapping by default as RFR does. Instead, it randomly selects values from the whole set each time. It then splits each node randomly, as opposed to RFR's method of splitting nodes by minimizing error. For optimization, ETR ultimately selects from the best estimator in the end, just as RFR does.

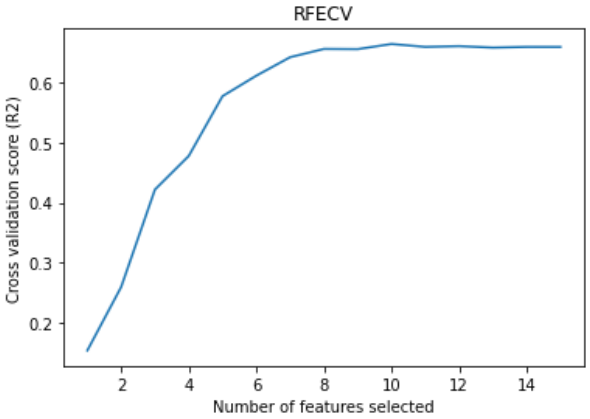


The ETR results are then used in the RFE wrapper, which removes the least important features and then reruns the estimator on the pruned features. It does this recursively until it reaches either a specified amount of features or until it has removed every feature.

I did not specify the desired number of features for our purposes. Instead, I passed them to the algorithm and analyzed the results.

Recursive Feature Elimination with Cross-Validation

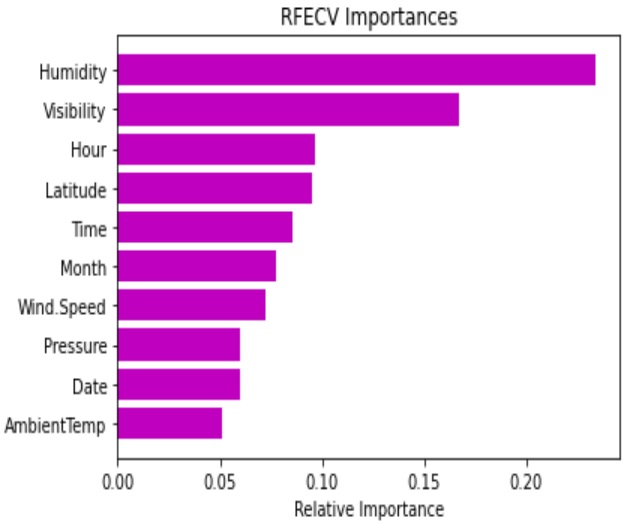
Following the RFE run, the RFECV algorithm uses the same default ETR. This method differs from the former because it adds cross-validation scores to determine the least important features and eliminate them recursively. Computational expense is raised, but it typically yields better results due to the added cross-validation step.



[12] (RFE)
[13] (ETR)

The graph immediately above plots the cross-validation scores for each amount of features used in the function. This shows that including more than 8 or 9 features, in this case, gains little. As to which features will comprise the optimal amount, we can use RFECV to rank them with the former RFE process.

RFECV results differ from those of the RFE. Most notably, the RFECV results suggest more total features, introducing “Wind.Speed,” “Pressure,” “Time,” and “Visibility” to the top rankings. Also, “Ambient.Temp” falls in rank to be superseded by others.



Which to Use

In selecting the best features for modeling, I chose to use the results from the RFCV procedure. The selections made with cross-validation are likely to be more valuable since they result from more extensive exposure to the training data.

Models

With an optimal selection of features determined, I trained various models to the training data with the selection of features. For comparison to the results of the tuned models, I initiated a baseline model using Sci-Kit Learn's Dummy Regressor function:

- **Dummy Regressor** (Dummy)
  (Baseline Model)

Using a baseline model affords a comparative perspective for the other models as it uses nothing more than a *mean* measurement to predict the target. It accounts for none of the intra-play of the data, so as a comparison, we can tell whether the other models have learned anything from the data.

The following models will be tuned and used for predictions:

1. **Random Forest Regression** (RFR)
2. **Stochastic Gradient Descent** (SGD)
3. **Multi-layer Perceptron regression** (MLP)

RandomizedSearchCV (RCV)

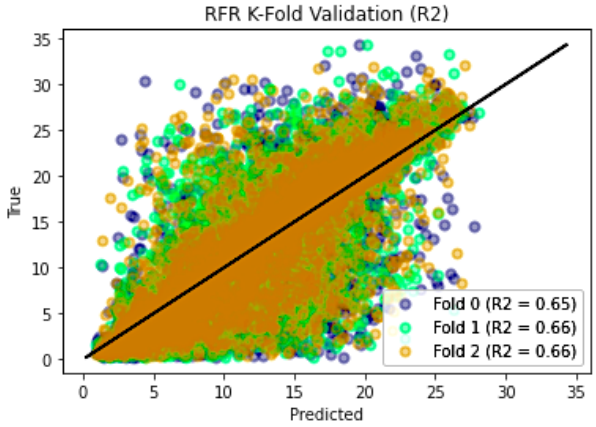
I tuned the hyperparameters of each model using RCV. This learner optimizes the parameters of a given model by varying its parameters over a specified amount of iterations. By fitting and scoring (CV) to the varying models, it determines which combination of parameters results with the least error.

1. Random Forest Regression (RFR)

RFR fits a specified amount of random forests to sub-samples of the data. I have chosen to start with the default number of these estimators: 100. The results are then averaged to construct the final estimator for the data. I have chosen the RFR because the sheer number of random trees should do well to cancel out one another's errors, producing a reliable and reliable model.

RCV: Best RFR Estimator

```
RandomForestRegressor(bootstrap=False, max_depth=30,
                      max_features='sqrt',
                      min_samples_split=10,
                      n_estimators=554))])
```

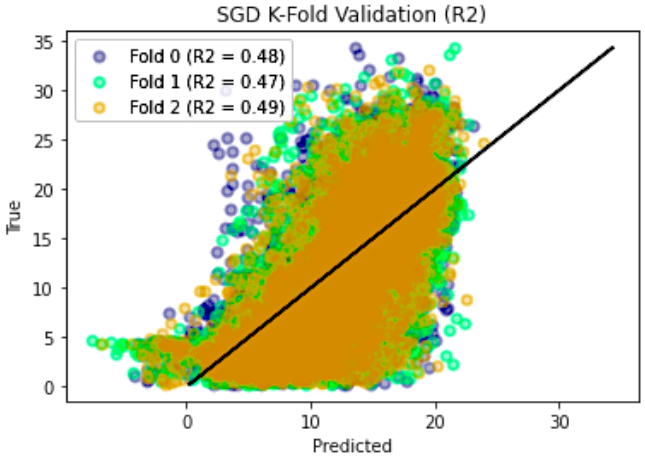


2. Stochastic Gradient Descent (SGD)

SGD uses gradient descent to perform linear regression on the data. The added benefit of choosing points on each iteration stochastically introduces randomness to the fitting and speeds up computation.

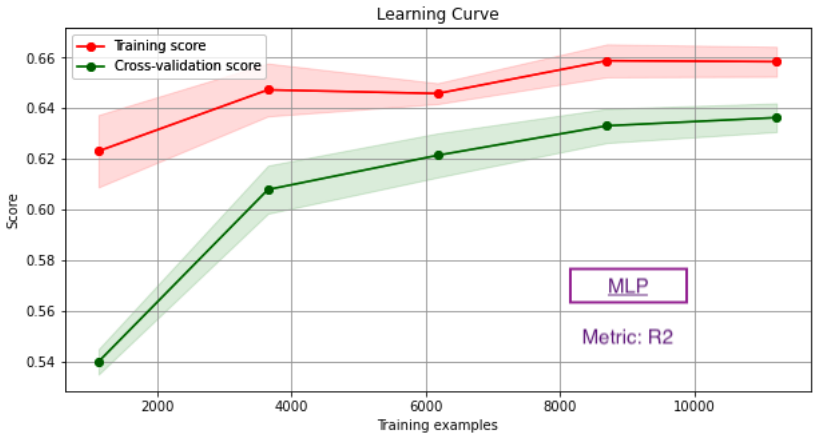
RCV: Best SGD Estimator

```
SGDRegressor(eta0=1, learning_rate='adaptive',
             loss='squared_epsilon_insensitive',
             max_iter=4500, penalty='elasticnet')
```



3. Multi-layer Perceptron regression (MLP)

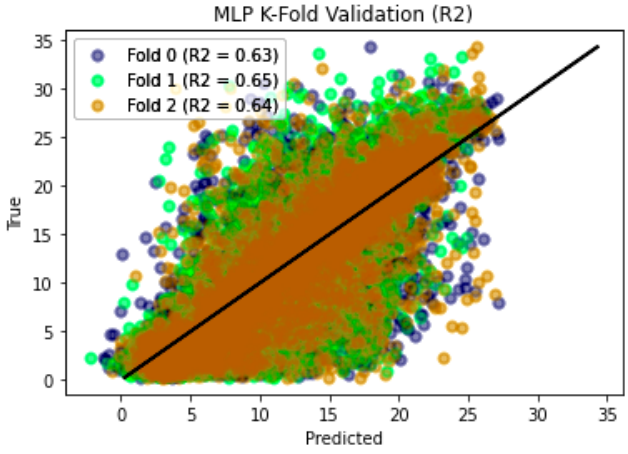
MLP is an artificial neural network (ANN) algorithm that optimizes the squared error in a regression problem by employing a specified solver. In this case, I have chosen to use MLP to introduce a non-linear approach to the modeling process to better account for any non-linear relationships within the data.



Unlike the other models, MLP's learning curve suggests that it may benefit from more data. Further discussion on this topic continues in the conclusory section below.

RCV: Best MLP Estimator

```
MLPRegressor(alpha=10, hidden_layer_sizes=(50, 100, 50),
             learning_rate='adaptive', solver='sgd')
```



Metrics

I have used the following metrics to score the models:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R2)

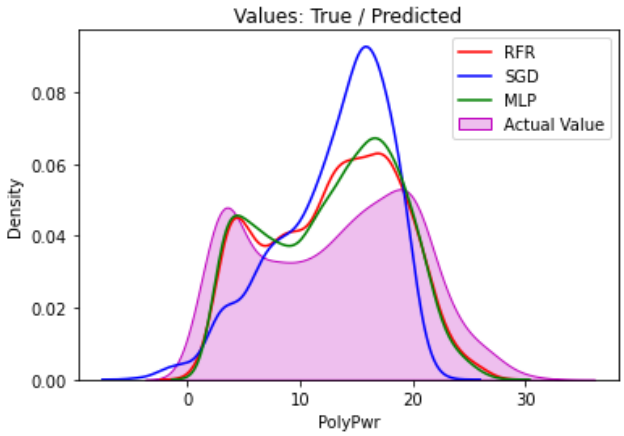
Results and Analysis

Each tuned model fitted to the holdout data (X_test). Once fitted, the models were used to generate predictions. Comparing these predictions to the true values of the target variable (y_test) resulted in the following error scores:

Model	MAE	RMSE	R^2
Dummy	6.167	7.142	0.0
RFR	2.616	3.985	0.689
SGD	4.034	5.158	0.478
MLP	2.892	4.208	0.653

When tested on the holdout data, each constructed model performed better than the dummy model. However, RFR tested better by every metric. MLP also performed considerably well, with an R^2 of only .036.

The figure on the right shows the density of each model's predictions. For example, the graph shows that most predictions fell between 10 and 20 watts. In the case of the SGD, more than 90% lie within this range. We can corroborate this by observing the mean predictions of each model:



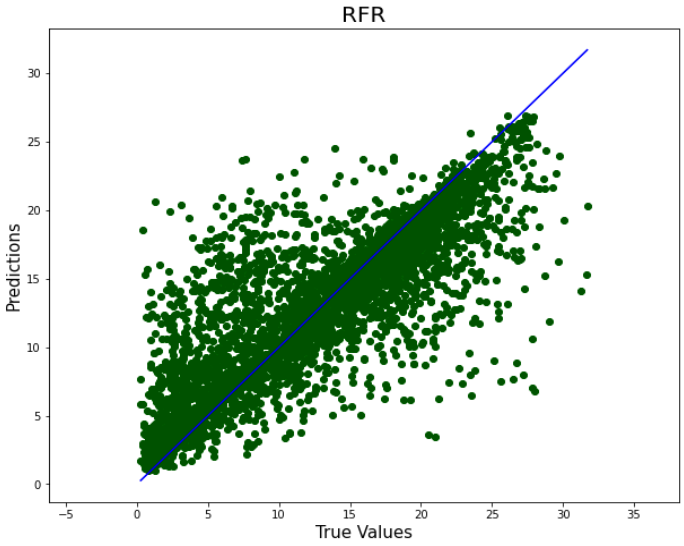
Mean Prediction:	RFR:	13.06
	SGD:	13.0
	MLP:	13.07
Mean of True values:		*12.95*

The RFR and MLP models captured more of the variance in the data of the holdout set than SGD. We see this, particularly, by the concentration of their predictions made within the >10 watts range. Like the distribution of the true values, RFR and MLP distributed their predictions bi-modally.

The fact that SGD's mean prediction came closer to the mean true value, despite capturing less of the variance, seems to reflect

SGD's bias toward Brownian motion[14]. This tendency toward a more Gaussian, or "normal," distribution appears ill-suited to the bi-modal distribution of the true values.

As we see in the plot on the right, the model fit well to the center of the data distribution overall yet didn't anticipate such a skew toward higher pwr output. The lack of generalizability shows here as it fails to account for the higher power values.



Random Forests models can distinguish subtle yet significant variations by feature and value. Models based on gradient descent may not attribute the same importance to particular variables. As seen in the RFR plot, our trained model can strike a more effective balance between the target variable's distribution modes, making a distinction that SGD did not.

[14] Şimşekli, Gürbüzbalaban, Nguyen, Richard, Sagun, "On the Heavy-Tailed Theory of Stochastic Gradient Descent for Deep Neural Networks," arXiv. eprint 1912.00018. Nov 2019

Conclusion

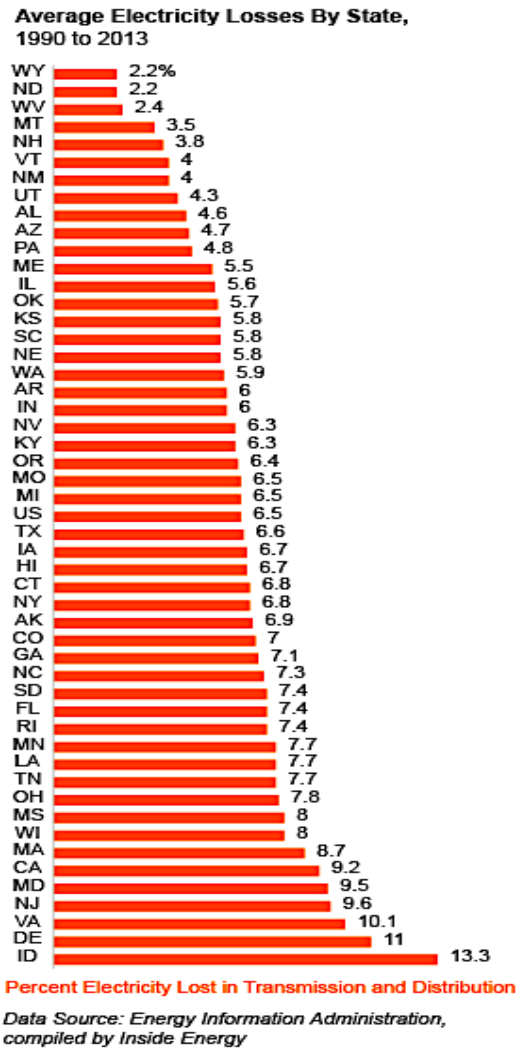
My goal for this project has been to engineer machine learning models that can effectively predict solar power potential, given a list of well-selected variables. To this aim, I began with an exploratory analysis of the data and used my findings, along with recursive feature elimination, to systematically select the most valuable features.

I applied RCV to tune each model's hyper-parameters with the selected list of features. Once adjusted, I fit them to the test data and predicted the final values of the target variable. I then evaluated the performance of the models using three standard regression metrics.

RFR owes much of its predictive accuracy to the feature selection process. Systematically determining the most valuable variables and dropping the others, particularly the location variable, has yielded an R2 score of .689. This score is a slight improvement over the "stacked model" covered in the related work cited, which achieved an R2 score of .681.

In concluding my analysis, this project presents a few potentially significant avenues for further inquiry. Any model predicting the potential for solar output must consider additional factors before it may be deployed. One of the most notable factors concerns the loss of electricity due to transmission and distribution (T&D).

The US grid loses an average of 5-6% of electricity in the US each year[15] due to T&D. As seen in the figure to the right[16], this loss percentage varies considerably by state, and higher loss rates occur in more densely populated, less rural states. In her report on grid loss, the Wirfs-Brock attributes this to the relatively lower-voltage capacity of urban power distribution and the fact that more rural states utilize



[15] EIA - US Energy Information Administration. Nov 2021.
[16] Wirfs-Brock, Jordan. "Lost In Transmission: How Much Electricity Disappears Between A Power Plant And Your Plug?" *Inside Energy*. Nov 2015.

higher-voltage lines that require a greater expanse of non-residential land.

Further modeling of potential solar output will be best contextualized within a data framework that accounts for the grid loss—optimizing the proximity of power collection infrastructure to the communities it serves.

Community Solar[17], which refers to developing local solar power infrastructure on the county and neighborhood level, offers a promising direction for continued study. By bringing clean energy collection closer to the stakeholders, increased community solar stands to reduce the loss associated with T&D. Perhaps, more importantly, the decentralization of the grid improves not only the resiliency of the grid but that of local communities as well through job increase and the opportunity for relatively more economic independence.

Further predictive analysis of solar potential should account for proximity to and the interests of local stakeholders within the context of community solar.

Thanks

to Michael Darling for his mentorship through the program so far.
to CodingNomads for teaching me.
to the open-source community for sharing so much.

[17] "An Opportunity to Enhance Sustainable Development on Landfills and Other Contaminated Sites." *EPA*. Dec 2016.