**Predicting solar power output using machine learning techniques**

Applying data science solutions to renewable energy challenges.



Photo by [Andreas Gücklhorn](https://unsplash.com/@draufsicht?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral" \t "_blank)

**Introduction**

Solar energy is one of the leading renewable energy sources in the world and it continues to grow. However, it depends on sunlight which is an intermittent natural resource. This makes power output predictability critical for the integration of solar photovoltaics into traditional electrical grid systems.

In the current analysis, power output from horizontal photovoltaics installed in 12 locations in the northern hemisphere is predicted. Only location and weather data are used without information about irradiance. While irradiance is a strong predictor of solar power output, collecting this information about a location is often tedious and its estimation may have significant errors. Hence, the ability to predict power output without irradiance data needs to be further explored to save time, effort, and cost with no significant loss of accuracy.

The data is publicly available on [Kaggle](https://www.kaggle.com/saurabhshahane/northern-hemisphere-horizontal-photovoltaic" \t "_blank) and consists of 14 months of power output, location, and weather data. In addition, notebooks used for this analysis are made available on [Github](https://github.com/abbey2017/solar-energy-analytics" \t "_blank). The remaining sections in this article will cover data processing, modeling, results, and conclusions.

**Data Processing**

Available variables in the data are explored, visualized, and pre-processed before being passed to the machine learning algorithms.

*Data exploration*

The dataset consists of 21,045 rows and 17 columns. Let’s explore the available columns in the dataset using functions in pandas- Python data analysis library.

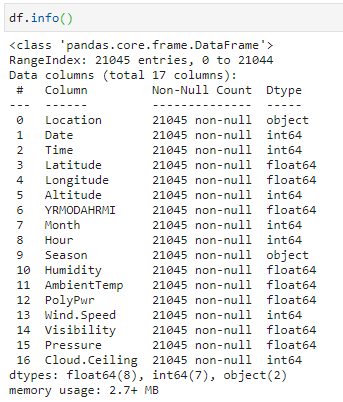


Image by author

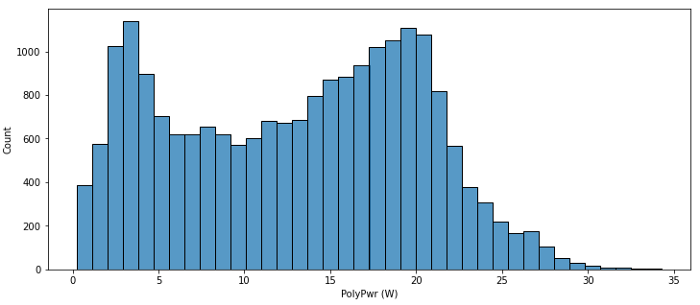
As shown above, there are no missing values in the dataset (yay!!). Also, the “YRMODAHRMI” column was dropped because it is not intuitive and no description is provided. The target variable is PolyPwr (power output).

In addition, the top five rows of the dataset are shown below:



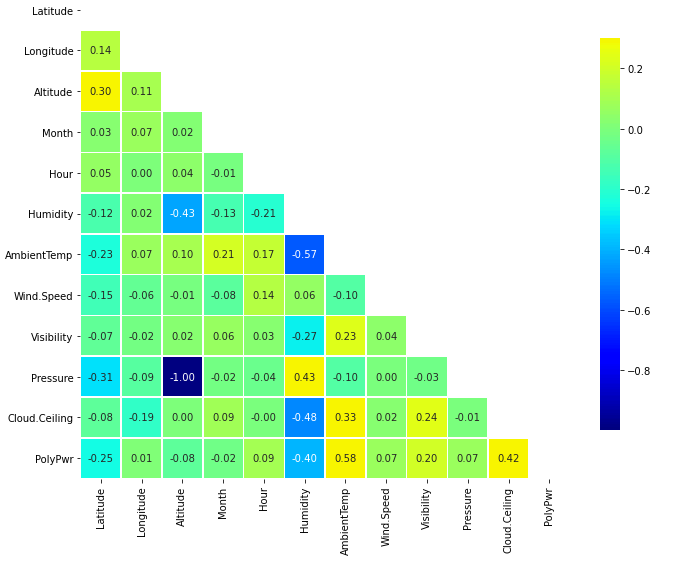
Image by author

Next, let’s inspect the distribution of the target variable. The histogram below does not show a significant skew although there is a limited representation of power output above 30 W. Hence, no additional difficulty is expected in predicting the target variable due to its distribution in the available dataset.



Histogram of power output variable in the entire dataset. (Image by author)

All is good so far. Now, let’s visualize the correlation between available features and power output.



Correlation between available features and power output. (Image by author)

From the correlation plot, ambient temperature, cloud ceiling, and humidity are the top three most correlated features with solar power output. It should also be noted that latitude has a significant correlation with power output while longitude does not show the same behavior. Hence, longitude was dropped from the modeling process. Altitude is also dropped because it has a perfect correlation with pressure but does not vary for a given location.

*Feature engineering*

Here, let’s create new features from existing ones to make categorical variables usable in our machine learning algorithms and also capture more patterns in the data.

First, we perform the encoding of categorical variables namely location and season using the one-hot encoding method.

# Encode location data  
df\_with\_location\_en = pd.get\_dummies(df, columns=['Location'], drop\_first=True)# Encode season data  
df\_with\_loc\_season\_en = pd.get\_dummies(df\_with\_location\_en, columns=['Season'], drop\_first=True)

Secondly, let’s create cyclic features using month and hour data. It should be noted that only data between 10 am and 3 pm is available which cuts out the period when the systems are not expected to generate power.

# Define time bounds in data  
min\_hour\_of\_interest = 10  
max\_hour\_of\_interest = 15# Calculate time lapse since onset of power generation  
df\_with\_loc\_season\_en['delta\_hr']= df\_with\_loc\_season\_en.Hour -  
 min\_hour\_of\_interest# Create cyclic month features  
df\_with\_loc\_season\_en['sine\_mon']=  
 np.sin((df\_with\_loc\_season\_en.Month - 1)\*np.pi/11)  
df\_with\_loc\_season\_en['cos\_mon']=   
 np.cos((df\_with\_loc\_season\_en.Month - 1)\*np.pi/11)# Create cyclic hour features  
df\_with\_loc\_season\_en['sine\_hr']=   
 np.sin((df\_with\_loc\_season\_en.delta\_hr\*np.pi/(max\_hour\_of\_interest  
 - min\_hour\_of\_interest)))  
df\_with\_loc\_season\_en['cos\_hr']=   
 np.cos((df\_with\_loc\_season\_en.delta\_hr\*np.pi/(max\_hour\_of\_interest  
 - min\_hour\_of\_interest)))

Additional correlation analysis including newly created features shows a perfect correlation between the cosine of date features and their actual values (Month and Hour). Hence, Month and Hour features are dropped in the modeling process.

The final list of features used for modeling is shown below:

selected\_columns = ['Latitude', 'Humidity', 'AmbientTemp',   
 'PolyPwr', 'Wind.Speed', 'Visibility',   
 'Pressure', 'Cloud.Ceiling', 'Location\_Grissom',   
 'Location\_Hill Weber', 'Location\_JDMT',   
 'Location\_Kahului', 'Location\_MNANG',   
 'Location\_Malmstrom', 'Location\_March AFB',   
 'Location\_Offutt', 'Location\_Peterson',   
 'Location\_Travis',   
 'Location\_USAFA','Season\_Spring',   
 'Season\_Summer', 'Season\_Winter', 'sine\_mon',   
 'cos\_mon', 'sine\_hr', 'cos\_hr']

*Data splitting*

The entire dataset is split into 80% training data and 20% test data. The test data is held out and unseen throughout the hyper-parameter tuning and training of the different models.

**Modeling**

Three models (Random Forest — RF, Light Gradient Boosting Machine — LGBM, and Deep Neural Network — DNN) and a stacked ensemble were developed and compared with a baseline (K Nearest Neighbors— KNN) model.

*Metrics*

The R-squared metric is the ultimate metric for selecting the best-performed model in this analysis. Other metrics useful for assessing the performance of selected models include root mean squared error (RMSE) and mean absolute error (MAE).

R-squared

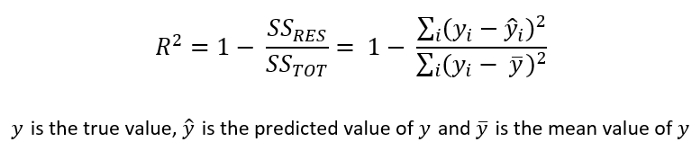


Image by author

RMSE

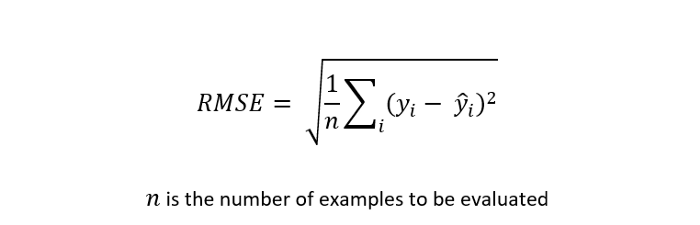


Image by author

MAE

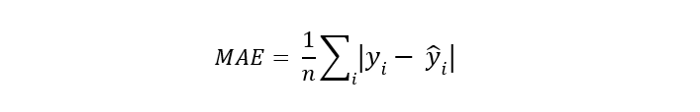


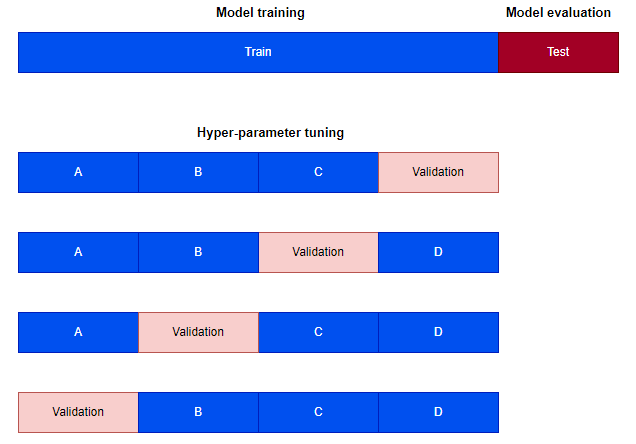
Image by author

The values of R-squared go from 0 to 1 and the higher the better while the values of RMSE and MAE have the same unit as the power output (W) and the smaller the better.

*Hyper-parameter tuning*

Each of the models was tuned using the [random search cross-validation](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html" \t "_blank) approach which enables the selection of the best combination of hyper-parameters based on the performance of the model on multiple splits of the training data.

In particular, 1000 permutations of the hyper-parameters were chosen and applied to 4 splits of the training data. The test data remains unseen and will be used for the final evaluation of the models chosen across different algorithms.

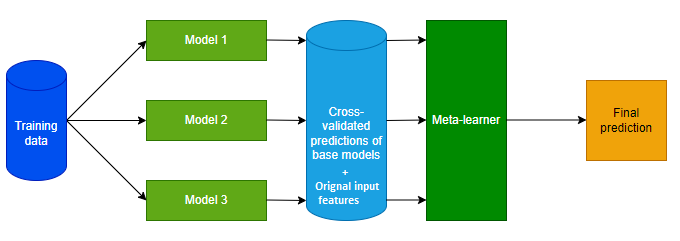


Four-fold cross-validation (Image by author)

*Model stacking*

Four disparate models (KNN, DNN, RF, and LGBM) were combined using the [stacking regressor module](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html" \t "_blank) in Scikit-learn- python machine learning library. A simple linear regression model was used as the meta-learner and it was trained on 4 fold cross-validated predictions of the base models as well as the original input features.

The stacking regressor uses the cross\_val\_predict function which returns for each example in the training data, the prediction that was obtained for that example when it was in the validation set. These predictions across the different base models are used as an input to the meta-learner (see Sklearn [user guide 3.1.1.2](https://scikit-learn.org/stable/modules/cross_validation.html" \l "cross-validation" \t "_blank) for details). This approach reduces the risk of overfitting.

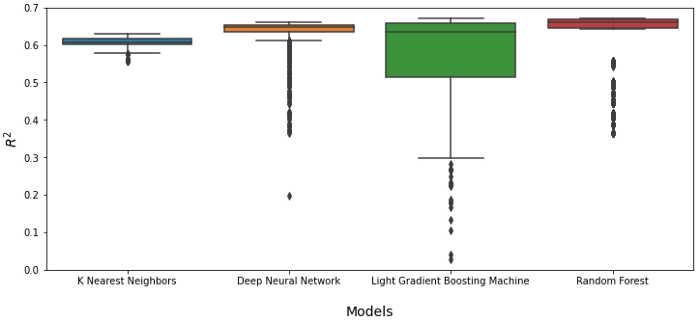


Model stacking (Image by author)

**Results**

*Cross-validation scores*

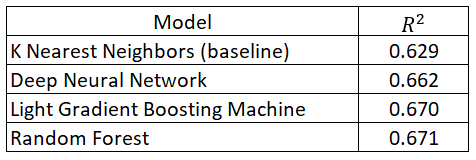
The cross-validation (CV) R-squared scores for 1000 random permutations of hyper-parameters for different algorithms are shown in the boxplot below:



CV scores of 1000 iterations of different algorithms. (Image by author)

As shown on the boxplot, the LGBM model is the most sensitive to hyper-parameters selection while KNN is the least sensitive.

Next, the best CV scores for each algorithm type is displayed below:

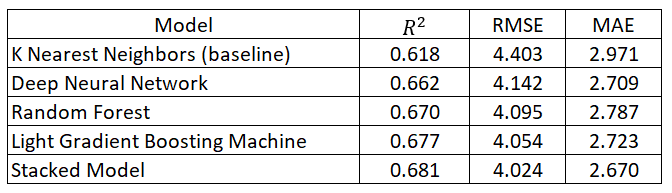


Four-fold CV best scores. (Image by author)

Results show that the best RF model has the highest CV score across all algorithms investigated.

*Test data scores*

The performance of each model is evaluated using the hold-out set which is 20% of the entire dataset. The results are summarized below:



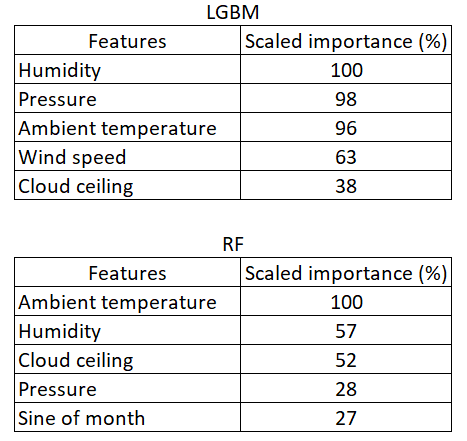
Model performance on the unseen test data. (Image by author)

The stacked model has the overall best performance with a 10% improvement compared with the KNN (baseline) model. In addition, the LGBM model is the best base model based on all metrics considered.

It should be noted that all models generalize well to the unseen test set with comparable performance between CV and test scores.

*Feature importances*

Using the LGBM and RF models with the capability to calculate feature importance, the table below shows the scaled importance of the top 5 features used in predicting the solar power output.



Top 5 scaled feature importance using tree methods. (Image by author)

Ambient temperature, humidity, cloud ceiling, and pressure are present in the top 5 features for both LGBM and RF models. The feature importance ranking obtained using the RF model agrees with the top 5 results reported in Pasion et. al., 2020.

**Conclusions**

Standard data science techniques have been applied to predicting the solar power output in 12 different locations. The results show that LGBM is the best base model while the stacked model gave the overall best performance.

The best model may be further trained for individual locations to achieve even better scores however, that is not covered in this analysis.

Finally, the approach followed in this work can be used as a standard procedure to solve predictive analytics problems in the renewable energy space.