INDUSTRIAL & SYSTEMS ENGINEERING
TEXAS A&M UNIVERSITY

ISEN 619 Project: Wind Turbine Power Prediction

Ahmadreza Chokhachian, Mohammad Rezaeifar and Youssef Hebaish
Industrial & Systems Engineering, Texas A&M University

Executive Summary

As part of the WeDoWind power curve modeling challenge, modeling wind turbine power curves is the key mission of this project. Wind farm data from the Penmanshiel and Kelmarsh wind farms are being utilized to improve wind energy output planning by creating a power curve model for quality prediction. The goal is to develop a statistical learning approach that can accurately estimate wind power using a collection of predictors.

As a preprocessing step, the authors have modified the format of the training datasets due to differences in the format of the "Time" input between the training and test datasets. The training data had a standard date and time format, while the test sets had separate columns for Day/Night and month. To address this, the authors transformed the Time column in the training sets to Day/Night and month format. They used -1/1 encoding for Night and Day and 1 to 12 for the months to avoid dealing with categorical data.

Following the preprocessing step, the authors applied various learning methods to train a model to predict power output for the test dataset. Several models from the DSWE library were used, such as *TempGP*, *SVM*, *KNN*, *AMK*, and *BART*. In addition, other learning methods that are not included in the DSWE library were used, such as: *RandomForest* and *XGBoot*. RandomForest and XGBoost are two ensemble-based learning methods that use trees as base models. While RandomForest uses bagging in model construction, XGBoost uses boosting. XGBoost is generally more accurate and computationally less expensive. This is reflected in how RandomForest is tuned in this study, where the authors choose a particular turbine to use in manually tuning the model hyperparameters. The results of this manual tuning is generalized and used across all turbines. Because XGBoost is faster than RandomForest, a more refined manual tuning is performed, where the number of trees is used to calculate the average validation RMSE across all turbine to choose the optimal. Validation RMSE is calculated by calculating RMSE between predictions made for 20% of the training dataset that was not used in training–validation set approach. Test RMSE (MAE) are 0.0262 (0.0146) and 0.0281 (0.016) for XGBoost and RandomForest, respectively.

While XGBoost and RandomForest performed signficantly better than the standard industry method, learning methods from the DSWE library did not perform as good. That is, based on test RMSE, SVM performed the best among DSWE learning methods with a test RMSE of 0.0289.

Other learning methods, such as DeepGP, ScaledVechia, and TempGP, are being studied and implemented, which are expected to provide competitive results. In addition, a more refined optimization of XGBoost hyperparameters is one direction where expansion is possible. Using a high-performance computer, performing a grid search would be feasible within a reasonable time.

1 Introduction

In this project, we model wind turbine power curves as part of the WeDoWind wind turbine power curve modeling challenge. Wind farm data from Penmanshiel and Kelmarsh wind farms are used to enhance wind energy production planning by developing a power curve model for quality prediction.

Power curve models are important in wind energy for several applications, both in the planning and operational phases of wind energy projects. Several data science/machine learning-based power curve models are being investigated, compared, and evaluated in our project. The aim is to get a statistical learning method that can perform a precise prediction on wind power using a set of predictors.

Binning is the standard method. The binning method is a statistical technique that groups a set of continuous or numerical data into a smaller number of "bins" or categories. This is done by dividing the range of the data into intervals of equal width and assigning each data point to the corresponding interval or bin. The binning method can also be useful for dealing with large data sets, as it reduces the amount of data that needs to be processed while retaining important information about the distribution of the data. However, it is important to choose the appropriate number and width of bins for the data in order to avoid losing too much information or introducing bias. An extension on binning can be using CART with only one dimension; in this case, the complexity of cart to standard binning will be having different width lengths.

The binning method works in a very simple way:

1. It takes only one most important variable and ignores the rest. Wind speed is the chosen predictor.

2. It takes the domain of wind speed and makes partitions with equal width sizes of 0.5 m/s.

3. Average all the power data and use the sample average as the representation of the power response for that specific bin;

4. To make predictions, it checks to see which bin the test data belongs to and will output the mean calculated for that bin in the previous step.

While the binning method can be a useful tool for simplifying the analysis of data, it also has some disadvantages that should be considered, such as Information loss, Bias, Sensitivity to outliers, Dependence on bin size and number, and Difficulty in interpreting results.

Referring to Figure 1, and considering the fact that binning is a model with the least complexity, we expect to get smaller errors by increasing the complexity of statistical learning methods. This might not always be true; after having a lower bound on the variation and bias of the models, adding more complexity may lead to overfitting. Still, someone may use deep learning methods, and more and more complexity helps them, but in this project, our focus is to find a statistical learning method with smaller errors in making predictions.

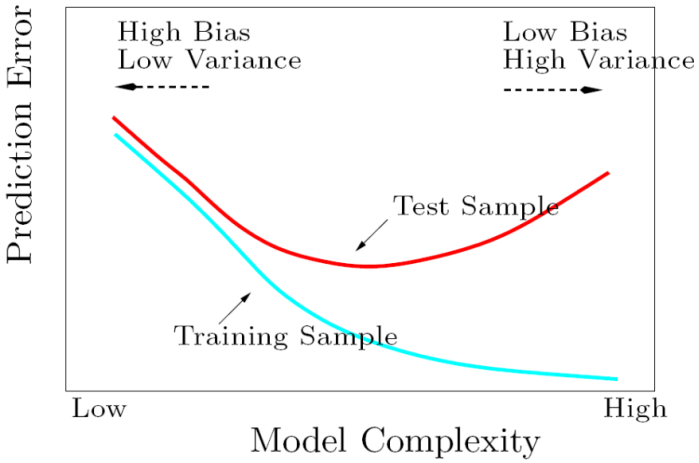


Figure 1: Model complexity and expected behavior of statistical learning methods

2 Methods & Models

In this section, we discuss our methodology in power curve prediction, including data preparation and model building. Data preparation includes all data cleaning or manipulation that preceded using the data model building and prediction. We also discuss the different models we used in our study, which include: RandomForest, XGBoost, SVM, and different model from the DSWE library. For each of the aforementioned models, we discuss how the model works, model building process, and model evaluation.

2.1 *Data Preparation*

The first challenge we faced before utilizing the data with statistical learning methods was the difference between the data structure of training and test datasets. Specifically, the "Time" input in training data was not in the same format as in the test datasets. Time was in the format of standard date and time in the training data; instead, in the test sets, we had two columns of Day/Night and month. Then we decided to transform the Time column to Day/Night and month in the training sets. In some methods like CART, the categorical inputs are applicable, but it is not true for all methods. Because of this, we decided to use -1/1 encoding for Night and Day and 1 to 12 for the months; this way, we did not have to deal with categorical data anymore. This might be criticized because of the curse of dimensionality; we agree that these predictors may not play a significant role in making predictions. We will let the dimension reduction approaches or subset selection algorithms decide it automatically because it is hard to judge which one out of the inputs, for example, month and wind speed sensor 1 in the existence of wind speed, has a more significant effect on output. This framework for the data is used in all models and methods we will discuss in the next sections.

2.2 *Models*

2.2.1 *RandomForest*

RandomForest is a popular machine learning algorithm that is widely used in predictive modeling. It belongs to the family of ensemble methods, which combine multiple models to generate more accurate predictions. In RandomForest, the model consists of an ensemble of decision trees, where each tree is trained on a random subset of the training data. One of the major advantages of RandomForest is its ability to handle high-dimensional data and complex relationships between variables. It can capture non-linear relationships and interactions between features, which makes it suitable for a wide range of prediction tasks. RandomForest is also robust to overfitting, which occurs when a model is too complex and fits the training data too closely. Another advantage of RandomForest is its ability to provide feature importance measures. This can be useful for understanding which features are most important for making predictions and can help with feature selection and model interpretation (Breiman, 2001).

In RandomForest, each decision tree is trained on a random subset of the training data and a random subset of the features. This process is repeated multiple times to create a set of decision trees. During prediction, the input data is passed through each decision tree, and the predictions from each tree are combined to generate a final prediction. To prevent overfitting, RandomForest uses two techniques: bagging and random feature selection. Bagging involves randomly sampling the training data with replacement to create multiple training sets. Each decision tree is then trained on a different training set. Random feature selection involves randomly selecting a subset of the features at each node in the decision tree. This helps to reduce the correlation between the decision trees and makes the model more robust to overfitting (Breiman, 2001).

On the other hand, RandomForest can be slow for large datasets, which is the case in this project. This pitfall is exacerbated by having 20 different turbines for which a model could be optimized. That is, RandomForest's hyperparameters are optimized using a grid search that evaluates different combinations of hyperparameters to evaluate the resulting error. Due to having many parameters, there are numerous combinations to test, resulting in an expensive grid search. Therefore, we opted to use only turbine in hyperparameters tuning and generalize those parameters across all locations. In addition, in training and validation, we train the model on 80% of the dataset, and estimate the validation error using the remaining 20% of the dataset. The training-validation split is done randomly to take into consideration the difference between the prediction period and the training period.

To reduce the time needed for hyperparameters tuning, default values of most hyperparameters are used in making predictions. For instance, the *max_depth* hyperparameter is used as default, which dictates that each tree will expand until all leaves are pure or until all leaves contain less than *min_samples_split* samples. On the other hand, *oob_score*, which controls whether to use out-of-bag samples to estimate the generalization score, is changed to True while the default value is False. The only hyperparameter that is tuned through a search is the number of trees. In addition, the number of trees is tuned using only one dataset, which is chosen to be *Kelmarsh_df1_training*.

Figure 2 shows validation RMSE for a range of the number of trees from 100 to 1000.

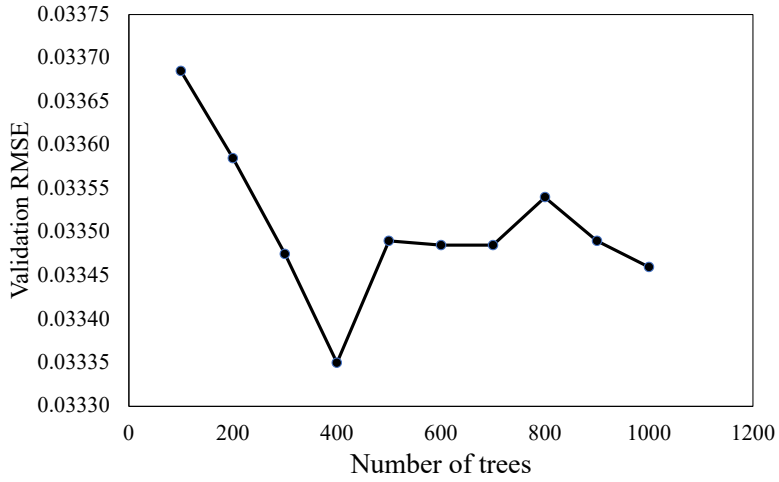


Figure 2: Validation RMSE as a function of the number of trees used on *Kelmarsh_df1_training*.

The optimal number of trees for this turbine is 400, and therefore, the chosen number of trees in making predictions for all turbines is 400.

The algorithm for the tuning procedure is as follows:

for $n = 100;\ n <= 1000;\ n = n + 100$ **do**
    x_train $\leftarrow$ 80% training data w/o target;
    y_train $\leftarrow$ 80% corresponding target;
    x_val $\leftarrow$ 20% remaining training data w/o target;
    y_val $\leftarrow$ 80% corresponding target;
    Train RF model using n trees and training subset;
    y_pred $\leftarrow$ prediction using x_val;
    RMSE_val $\leftarrow$ RMSE(y_pred, y_val);
end

2.2.2 *XGBoost*

XGBoost (Extreme Gradient Boosting) is a commonly used machine learning method in predictive modeling. It belongs to the family of gradient boosting techniques, which combine weak learners repeatedly to increase model performance. One of XGBoost's key features is its capacity to handle enormous datasets and high-dimensional data. It is extremely scalable, having the ability to handle millions of data points and hundreds of characteristics. XGBoost also offers significant regularization capabilities, which can aid in the prevention of overfitting and the improvement of generalization performance. XGBoost also has the capacity to handle a broad range of data formats, including numeric, categorical, and text data. It can also manage missing data and feature scaling automatically, saving time during data preparation (Chen and Guestrin, 2016). XGBoost is also well-known for its speed and precision. On a variety of benchmark datasets and real-world applications, it has been shown to outperform other popular machine learning algorithms such as Random Forest

and Support Vector Machines.

XGBoost works by building an ensemble of weak learners, typically decision trees, in a greedy and sequential manner. Each weak learner is trained to minimize the loss function, which measures the error between the predicted and actual values. During each iteration, XGBoost adds a new weak learner to the ensemble that reduces the loss function the most. To prevent overfitting, XGBoost employs regularization techniques, such as L1 and L2 regularization, and also includes a parameter called the learning rate, which controls the contribution of each weak learner to the final prediction. XGBoost also includes features such as early stopping, which can automatically stop the training process when the model starts to overfit.

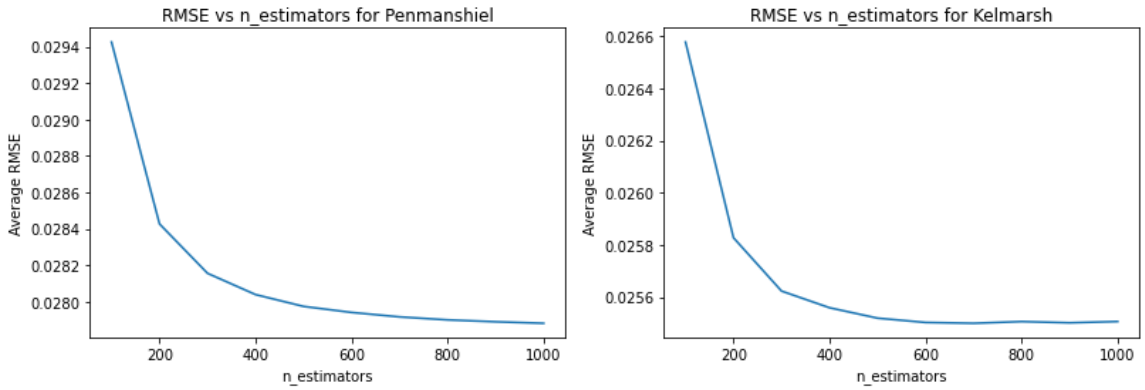
Moreover, XGBoost is much faster than RandomForest, which allows us to better tune it. However, tuning an individual model for each turbine is, nonetheless, computationally expensive. The reason behind this is that there are numerous combinations of hyperparameters to try for each individual model. For instance, if we consider the possible values of the hyperparameters shown in Table 3, we have 288 combinations. Although, each iteration—training and producing predictions—takes around 45 seconds, the total time needed to iterate over all possible combinations for all 20 turbines is 72 hours. This time is estimated based on a regular computer's performance. Therefore, this time is expected to be much less if a high-performance computer is used. For that reason, we resort to manually tuning the hyperparameters, one at a time, to reach a reasonable value for validation RMSE.

Parameter	Values
n_estimators	$[400, 600, 800, 1000]$
learning_rate	$[0.05, 0.1]$
max_depth	$[6, 7, 8, 9]$
subsample	$[0.7, 0.8, 0.9]$
colsample_bytree	$[0.7, 0.8, 0.9]$

Table 1: Possible XGBoost hyperparameters values that could be used in a grid search

Although hyperparameters are manually tuned, we determine the used number of trees by iterating over 10 possible values ranging from 100 to 1000 with an increment of 100. Figure 3 shows the average validation RMSE across all turbines for Kelmarsh and Penmanshiel as a function of the tree size. The other hyperparameters values are as follows:

- *'learning_rate'*: 0.05
- *'max_depth'*: 8
- *'subsample'*: 0.7
- *'colsample_bytree'*: 0.9



(a) Penmanshiel (b) Kelmarsh

Figure 3: Average validation RMSE for (a) Penmanshiel and (b) Kelmarsh using different tree size.

To choose the number of trees, we used a heuristic approach known as the elbow method, where we select the value at which we start to observe a diminishing return in terms of gain. That value is 400 trees. When the number of trees is larger than 400 the percentage reduction in validation RMSE is not significant, and therefore, we choose 400 as the value for *'n_estimators'* parameter.

2.2.3 *Model Selection*

In most methods we implemented to compare the models or to tune the parameters, we were using 10-fold cross-validation or 80-20 random split after randomly reordering the data. The reason for this step was to prevent losing whole data from a specific duration in the fit. For example, if the 80-20 split is implemented without randomization, the fit to the 80 percent of the data may lose one full month, like December; And the test error will be on December's data, which due to seasonal effects, may be biased from the model we already trained. To prevent bias, we always randomized the data before using an 80/20 split or 10-folds.

2.2.4 *DSWE Library*

DSWE library in R was given as a possible reference for the project; we decided to run some of the methods in this package to get a general baseline for other models we wanted to use. From DSWE, we utilized BART, SVM, KNN, TempGP, and AMK. The methods were not competitive, so we skipped discussing these algorithms. The results of these methods are ordered in Table 2 from the aspect of MAE and in Table 3 from the RMSE point of view.

Methods	MAE
TempGP	0.0162
BART	0.0167
SVM	0.0171
AMK	0.0199
KNN	0.0264

Table 2: DSWE methods sorted based on MAE

Methods	RMSE
SVM	0.0289
BART	0.0295
TempGP	0.0300
AMK	0.0325
KNN	0.0416

Table 3: DSWE methods sorted based on RMSE

3 Results

In this section, we highlight the results of some models discussed in Section 2. Models are evaluated using RMSE and MAE calculated using test datasets that were not included in the training process, and for which we do not have the target values. Tables 4 and 5 show test RMSE and MARE of the top 4 performing learning methods, measured by their test RMSE.

Rank	Learning method	RMSE
1	*XGBoost*	0.0262
2	*RandomForest*	0.0281
3	*TempGP*	0.0289
4	*SVM*	0.0300

Table 4: Test RMSE of the top performing 4 learning methods.

Rank	Learning method	MAE
1	*XGBoost*	0.0146
2	*RandomForest*	0.0160
3	*TempGP*	0.0171
4	*SVM*	0.0162

Table 5: Test MAE of the top performing 4 learning methods.

4 Conclusion

As we discussed in the results table among current models, XGBoost is outperforming. Our future investigation includes attempting a set of methods based on Gaussian Processes. We expect to get competitive results from these methods because they are able to provide posterior distribution of the fit. For now, DeepGP, ScaledVecchia, and TempGp are being implemented by our team, but technical issues did not let us get the final results and publish them in this report. For DeepGp, the package holder is Annie Sauer; her first guess was that the errors might be because of the order of data; the maximum dimension of locations she has tried up to now is 7, that way, we may perform dimension reduction before using deepGP.

In the future, several directions could be explored. We are particularly interested in performing a grid search over the hyperparameters values for XGBoost method. Performing auto-tuning, through a grid search, using a high-performance computer could significantly improve prediction power since each turbine would use its optimal hyperparameters values. Moreover, performing feature selection could significantly enhance the prediction power of different learning methods (using the vital few).

References

Breiman, L. (2001). Random forests. *Machine learning 45*, 5–32.

Chen, T. and C. Guestrin (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA, pp. 785–794. ACM.