International Journal of Forecasting 30 (2014) 395–401

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

A feature engineering approach to wind power forecasting GEFCom 2012

Lucas Silva

DTI Sistemas, Belo Horizonte, Brazil

ARTICLE INFO

Keywords:
GEFCom
Feature engineering
Gradient boosted decision trees
Linear regression
Machine learning
Time series

ABSTRACT

This paper provides detailed information about team Leustagos' approach to the wind power forecasting track of GEFCom 2012. The task was to predict the hourly power generation at seven wind farms, 48 hours ahead. The problem was addressed by extracting time- and weather-related features, which were used to build gradient-boosted decision trees and linear regression models. This approach achieved first place in both the public and private leaderboards.

1. Introduction

The "GEFCom 2012—Wind Forecasting" competition[1] posed the challenge of forecasting the hourly wind power generation for seven wind farms. A dataset containing historical power measurements for these wind farms, as well as meteorological forecasts of the wind components at the level of those wind farms, was provided. A detailed description of the dataset is provided by Hong, Pinson, and Fan (2013).

1.1. Challenges

A big challenge in this task was dealing with the time series nature of the dataset. Since no time series specific model was used, it had to be kept constantly in mind.

Another difficulty was in creating efficient features. Feature creation is one of the most important steps when solving a supervised learning problem. In order to do this, many features were derived from the dataset. The feature creation step had two main guiding principles:

1. Model the wind power generation equation, based on constants, the wind strength and direction, and the air density (surrogated). These features represent the windmill behavior.
2. Discover the relationship between the wind power generation at T and $T \pm n$. The objective here was to reflect the time series nature of the dataset, since the modeling techniques are not time series specific.

1.2. Dataset

As can be seen, the data provided (see Hong et al., 2013) consisted only of a series of hourly wind power generation values for each farm, and the forecasted wind strength and direction, issued every 12 h. This was done to mimic the real operation conditions, and did not include all of the desired information, such as the forecasted temperature and air density. Thus, to make it up for this missing information, many surrogate features were created, as will be explained later. Also, an important step was to build a consistent validation set using only the training period. This validation set allowed the construction of a model that did not overfit the training data.

1.3. Techniques

In this work, three main machine learning techniques have been used, all of them in the R statistical environment. Table 1 gives a brief description of how each algorithm was employed.

1.3.1. Linear regression

The linear regression method has been used in this work as a pre-processing step for combining the wind strength

E-mail address: lucas.eustaquio@gmail.com.

[1] http://www.kaggle.com/c/GEF2012-wind-forecasting.

0169-2070/$ – see front matter
http://dx.doi.org/10.1016/j.ijforecast.2013.07.007

Table 1
Techniques used.

Technique	Used for
Multiple linear regression (Geyer, 2003)	Influence of the wind strength and direction components for each farm Ensemble Post process to smooth the predictions (high frequency filter)
K-MEANS[a]	Similarity model to detect farm and overall behavior
GBM[b]	Models by farm Models by time slot Overall models

[a] R K-Means package, http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html.
[b] gbm: Generalized boosted regression models. http://cran.r-project.org/web/packages/gbm/index.html.

and direction components in one single feature. It has also been used to combine the models trained with GBM, and finally as a post-processing step to smoothen the moving average of the predicted values.

The linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to the observed data.[2] Eq. (1) shows the formal definition:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon, \quad (1)$$

where:

- $i = 1, \ldots, n$ is the training instance index;
- β_p are the regression coefficients;
- x_{ip} are the features; and
- ε is the residual error.

1.3.2. K-Means

In this work, K-Means was used to create similarity features. The objective of these features was to cluster instances that should have similar behaviors.

K-Means is a simple learning algorithm for clustering analysis. The goal of the K-Means algorithm is to find the best division of n entities into k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized. The pseudo-code for the K-Means algorithm is given in Fig. 1.[3]

1.3.3. GBM—generalized boosted models

GBM was used to train the intermediate models. The GBM package in the R environment with the Gaussian distribution function (Ridgeway, 2013) was used for this.

The R GBM package implements boosting for models, which is commonly used in statistics. The pseudo-code for GBM algorithm is shown in Fig. 2 (see Friedman, 2001). In Fig. 2,

- $F(x)$ is the function to minimize;
- $L(y, p)$ is the loss function;
- y is the output;
- p is the prediction;
- M is the number of trees; and
- $h(x; a)$ is a "weak learner" or "base learner", usually trees.

[2] http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm.
[3] Data Mining Algorithms in R/Clustering/K-Means. http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)
Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

```
foreach c_i ∈ C do
    c_i ← e_j ∈ E (e.g. random selection)
end
foreach e_i ∈ E do
    l(e_i) ← argminDistance(e_i, c_j) j ∈ {1...k}
end

changed ← false;
iter ← 0;
repeat
    foreach c_i ∈ C do
        UpdateCluster(c_i);
    end
    foreach e_i ∈ E do
        minDist ← argminDistance(e_i, c_j) j ∈ {1...k};
        If minDist ≠ l(e_i) then
            l(e_i) ← minDist;
            changed ← true;
        end
    end
    iter++;
until changed = true and iter ≤ MaxIters ;
```

Fig. 1. K-Means algorithm.

Algorithm 1: Gradient_Boost

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\tilde{y}_i = -\left[\frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$, $i = 1, N$
4. $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

Fig. 2. Boosting algorithm.

1.4. Approach overview

The basic framework of the approach is shown in Fig. 3. First, in the preprocessing step, features were created. Next, three types of models were trained using the processed data:

- models that were trained for each farm;
- models that were trained for each predicted time slot (1–3 h ahead, 4–6 h ahead, . . . , 45–48 h ahead); and
- overall models trained without splitting the samples (except for the cross-validation).

1.5. Organization of this paper

In this paper, the features and models used are described in detail. Section 2 describes the approach taken, explains the feature creation and modeling techniques

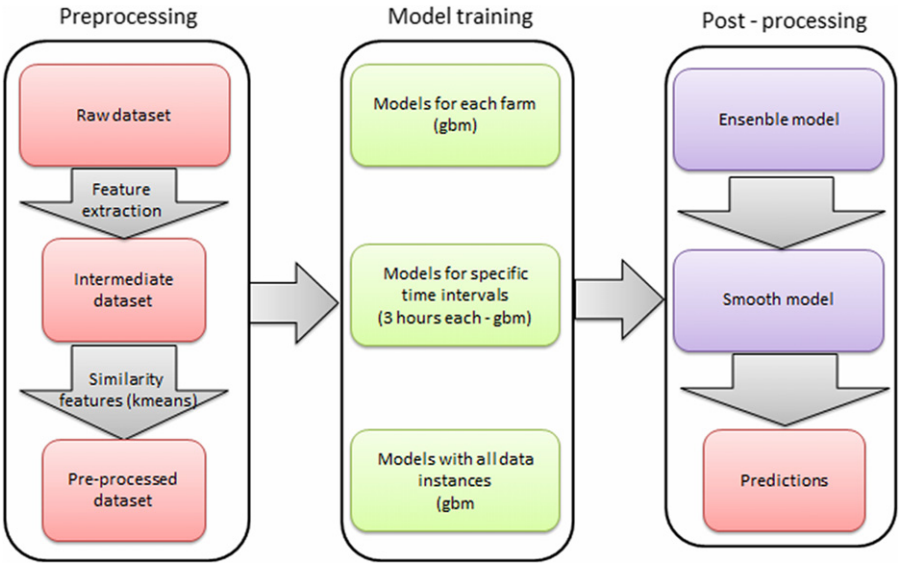

Fig. 3. Basic approach framework.

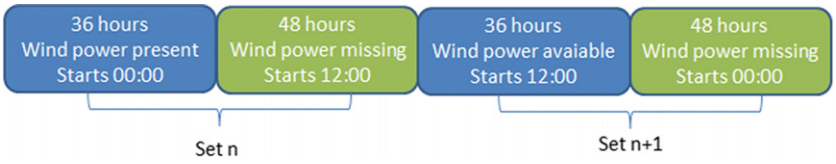

Fig. 4. Training/validation splitting.

used, and shows how the dataset was split for model training. Section 3 presents the results and discusses the aspects of the solution, and Section 4 presents the conclusion.

2. The approach

2.1. Validation set

The first step of the approach used was to build a consistent validation set. It was built in an attempt to replicate the structure between filled and missing spots found in the evaluation period. In order to replicate this structure, the training and evaluation periods were each split in 312 sets of 84 consecutive hours ($36 + 48$), and these sets were used to define the 5-fold validation splitting. Each training/validation fold assigned entire 84 h buckets to either training or validation.

Fig. 4 shows how the set assignment was done. One can see that the odd numbered sets start at 00:00 and the even numbered sets start at 12:00. This also happens with the missing spots in the evaluation period. As for the 5-fold splitting, it was done like this:

- Validation folds: Fold 1 contained sets 1, 6, 11, and so on, as the validation set, and the remaining sets as the training set. Fold 2 contained sets 2, 7, 12, and so on, for validation, and the other sets for training. Analogous splits were done for folds 3, 4 and 5.
- Test fold: All of the available filled data were used for building models in order to predict the missing data.

In this scenario, each model needed to be trained six times: once for each validation fold, in order to tune the model parameters and train the ensembles, totalling five times, and another time with all of the available data to predict values for the leaderboard. This procedure allowed us to produce consistent results most of the time, meaning that the improvements found in the available data almost always generalized to the missing unknown data (the leaderboard data).

In this problem, there is a time component in the data. Thus, instance-based random cross-validation tends to overestimate the models' performances, as it implicitly gives the model more information on the surroundings of each 48 h spot. This was verified in an early step benchmark, and is the reason why this validation scheme was used.

2.2. Feature creation

The first step in the creation of features was to create training instances. The dataset provided did not contain these explicitly. To create these instances, the files "train.csv" and "benchmark.csv" were processed, creating one entry for each date and each farm. The features hour, month and year were also created at this point. Fig. 5 shows what the data which were produced looked like, and Table 2 explains each feature.

The next step was to process the forecasts. Every forecast for each farm was used in this step. Each time-

Table 2
Features created in the first step.

Name	Type	Description
date	Date and time	This is the date and time of each wind power measurement; it is used mainly to join different tables of features
farm	Categorical (1–7)	Represents each farm
wp	Numeric	Wind power—value to be predicted
hour	Categorical (01–24)	Hour of each wind power measurement
month	Categorical (01–12)	Month of each wind power measurement
year	Categorical (2009–2012)	Year of each wind power measurement

Table 3
Forecast features.

Name	Type	Description
date	Date and time	Target forecast date
farm	Categorical (values 1–7)	Represents each farm
start	Date and time	Date when the forecast was issued
dist	Categorical (values 01–48)	Difference in hours of start and date; distance of the forecasting
turn	Categorical (00 and 12)	Indicates the starting hour of forecasting
set	Categorical (1–312)	Each set represents a period of 36 h + 48 h; the variable was used to do cross-validation training
ws	Numerical	Forecasted wind strength
wd	Numerical	Forecasted wind direction
wd_cut	Categorical ([0, 30] . . . (330, 360])	Categorical version of wind direction

Table 4
Historical features.

Name	Type	Description
start	Date and time	Date when the forecast was issued
farm	Categorical (1–7)	Wind farm
dist	Categorical (01–48)	Difference in starting hours and date; distance of the forecasting
wp_hnXX (XX = [01, 06])	Numerical	X previous known value: 01 is the value for the farm at the start date and time, 02 is the start time −1 h, and so on

```
                    date farm    wp hour month year
     1: 2009-07-01 00:00:00    1 0.045   00    07 2009
     2: 2009-07-01 00:00:00    2 0.233   00    07 2009
     3: 2009-07-01 00:00:00    3 0.494   00    07 2009
     4: 2009-07-01 00:00:00    4 0.105   00    07 2009
     5: 2009-07-01 00:00:00    5 0.056   00    07 2009
    ---
183711: 2012-06-28 12:00:00    3    NA   12    06 2012
183712: 2012-06-28 12:00:00    4    NA   12    06 2012
183713: 2012-06-28 12:00:00    5    NA   12    06 2012
183714: 2012-06-28 12:00:00    6    NA   12    06 2012
183715: 2012-06-28 12:00:00    7    NA   12    06 2012
```

Fig. 5. Instance creation, step 1.

stamp had four distinct forecasts, except for the ones in the leaderboard set, which had only the forecasts that would be available at the time of prediction. Some other features were also derived in this step. Fig. 6 and Table 3 show the data and the meaning of the features.

The next features created were the historical ones. For each forecast, the six values before the start of the forecast were used. In the test set, many of those values were unknown, and those instances were discarded during part of the training. For validation purposes, it would be useful to calculate those previous values for every single instance, even in the training set, as they would be known in a real situation. The historical features are given in Table 4.

The next features aimed at unifying the wind strength and wind direction in one single numeric variable. This unification was exploited later in the construction of a similarity model and the correction of predictions using moving averages. For wind power generation, it is expected that not only will the strength of the wind matter, but also its direction. The windmills will have different levels of efficiencies for different directions. The wind power conversion is proportional to both constants and ws^3 (cubic wind strength). It also depends on the air density.

```
                    date farm               start dist turn set   ws     wd    wd_cut
     1: 2009-07-02 13:00:00    1 2009-07-01 00:00:00   37   00   1 1.76 198.89 (180,210]
     2: 2009-07-02 13:00:00    1 2009-07-01 12:00:00   25   12   1 1.46 232.75 (210,240]
     3: 2009-07-02 13:00:00    1 2009-07-02 00:00:00   13   00   1 1.51 191.55 (180,210]
     4: 2009-07-02 13:00:00    1 2009-07-02 12:00:00   01   12   1 2.22 169.75 (150,180]
     5: 2009-07-02 13:00:00    2 2009-07-01 00:00:00   37   00   1 1.22 104.02  (90,120]
    ---
576572: 2012-06-28 12:00:00    3 2012-06-26 12:00:00   48   12 312 4.21  53.80   (30,60]
576573: 2012-06-28 12:00:00    4 2012-06-26 12:00:00   48   12 312 3.05 318.31 (300,330]
576574: 2012-06-28 12:00:00    5 2012-06-26 12:00:00   48   12 312 1.66 115.34  (90,120]
576575: 2012-06-28 12:00:00    6 2012-06-26 12:00:00   48   12 312 3.49 325.47 (300,330]
576576: 2012-06-28 12:00:00    7 2012-06-26 12:00:00   48   12 312 4.06 328.97 (300,330]
```

Fig. 6. Features created using forecast.

Table 5
Features unifying *ws* and angle.

Name	Type	Description
date	Date and time	Target forecast date
farm	Categorical (1–7)	Wind farm
dist	Categorical (01–48)	Difference in hours of start and date
wp	Numeric	Wind power forecast
ws.angle	Numeric	Influence of both *ws* and *wd* for the date
*ws.angle.p*3	Numeric	*ws.angle* for date −3 h. If not found, defaults to the nearest *ws.angle*
*ws.angle.p*2	Numeric	*ws.angle* for date −2 h. If not found, defaults to the nearest *ws.angle*
*ws.angle.p*1	Numeric	*ws.angle* for date −1 h. If not found. defaults to the nearest *ws.angle*
*ws.angle.n*3	Numeric	*ws.angle* for date +3 h. If not found, defaults to the nearest *ws.angle*
*ws.angle.n*2	Numeric	*ws.angle* for date +2 h. If not found, defaults to the nearest *ws.angle*
*ws.angle.n*1	Numeric	*ws.angle* for date +1 h. If not found, defaults to the nearest *ws.angle*

Table 6
Similarity features.

Name	Type	Description
date	Date and time	Target forecast date
farm	Categorical (1–7)	Wind farm
dist	Categorical (01–48)	Difference in hours of start and date; distance of the forecasting
clust.pos	Categorical (1–12)	Position of the forecast inside the 12 sequential values. The same as the remainder of the division of (dist − 1) by 12
begin	Categorical	Hour of the first value of the 12 h sequence
clust.farm	Categorical ($42 = 7 \times 6$)	Cluster that the 12 sequential hours got assigned to
clust	Categorical (1–24)	General cluster

This feature generation consisted in the construction of a simple regressive model for each farm. To avoid overfitting, 5-fold cross-validation was used. A very important issue in this cross validation is that it must use the set features calculated earlier. Each set must be either fully included or fully excluded. Doing instance-based sampling would lead to overfitting, because the regression will infer information related to each "36 h + 48 h" period.

The R formula used in this training was Eq. (2). However, *wd_cut* is not exactly the same as the earlier description. For this training, it was a categorical value that split *wd* in intervals of eight degrees (this splitting was found iteratively). The result of this training was called *ws.angle*. Another feature map was then built with these values. In this map, the three previous and three next values of the *ws.angles* forecasts were also included. The results are shown in Table 5.

$$wp \sim wd\_cut * (ws + ws \wedge 2 + ws \wedge 3). \quad (2)$$

Finally, the last features were similarity ones. Since the predicted values are a time series, some correlations between consecutive predicted values are expected. One way to let the model learn this was done in the last step. Including the previous and next expected values will make the model learn some level of dependency between $T \pm n$ and T. Another possible way to do this is via the similarity: extract the farm behavior for a set of predictions.

The similarity features were calculated using K-Means. For each 12 h forecasting period ([1 h–12 h], [13 h–24 h], [25 h–36 h], [37 h–48 h]), an instance was built. Six clusters were calculated for each farm, and 24 for a dataset including all farms. See Table 6 for the results.

The final features were obtained by joining all of the features which have been produced. To join the features, simply match the common columns and do a Cartesian product with the remaining. Taking Tables 2 and 6 as an example, we can see that the common features in them are *date* and *farm*. To join these tables, match those fields by their values. In this case, for each matching value in Table 2, four records will be found in Table 6, because each *wp* forecast was issued four times. The joining result will be obtained appending the Table 2 features to each match in Table 6 (Cartesian product).

Some features refer to values in the future (date + *n* hours). It is important to point out that those values are the forecasted ones, so the whole feature engineering here only uses data which are actually available at the time of forecasting.

2.3. Modeling techniques

All of the models were trained using the same 5-fold validation scheme. One very important characteristic of this training is that the validation sets were built to either include or remove the 36 h+48 h hour blocks as a whole, as has been said repeatedly (this is really important). Instance by instance sampling would reflect neither the real situation nor the test set one. By doing this, a consistent validation set was obtained. Significant improvements almost always led to improvements in the leaderboard.

The models were trained with two aims, as has been mentioned.

- To model the wind power generation equation, based on constants, wind strength, direction and air density. The last of these was surrogated by seasonal features, like time, hour and year. These factors express the windmill behavior.
- To discover the relationship between the wind power generation at T and $T \pm n$. The objective here was to introduce the time series nature of the models.

To achieve this, the following types of models were trained.

1. GBM with a Gaussian distribution and Eq. (3). The aim of this model was to investigate the inertial behavior and discover something about the effect of air density and other occasional environment influences. The inertial behavior was learned through $ws.angle(T \pm 3)$. Occasional influences were inferred using the *hour*, *month* and *year* categorical variables. It is possible that the use of temperature and/or humidity measurements could achieve better performances in real-world situations. The hour and month are related to the night/day and seasons relationships. The *farm* value will deal with any farm specifics, and *dist* with errors relating to the distance of the forecasts (the longer the distance, the greater the error).
2. GBM with a Gaussian distribution and Eq. (4). As before, but including the latest known value of the wind power. This value can provide some valuable contextual information and a starting point for the time series. The reason why both models exist is because many instances in the testing period were discarded in order to train this one (they lack wp_hn01). In real life situations this would always be available.
3. GBM with a Gaussian distribution and Eq. (5). This model was built in order to investigate environmental influences. Mixing these features with the previous ones did not show any improvement; however, doing separate trainings did.

$$wp \sim ws + farm + dist + ws.angle + ws.angle.p1 + ws.angle.p2 + ws.angle.p3 + ws.angle.n1 + ws.angle.n2 + ws.angle.n3 + hour + month \quad (3)$$

$$wp \sim ws + farm + ws.angle + ws.angle.p1 + ws.angle.p2 + ws.angle.p3 + ws.angle.n1 + ws.angle.n2 + ws.angle.n3 + hour + month + year + dist + wp\_hn01 \quad (4)$$

$$wp \sim farm + dist + wp\_hn02 + wp\_hn03 + wp\_hn04 + hour + month + clust.farm + clust + begin + clust.pos. \quad (5)$$

All of the models which have been mentioned were trained in three flavors: per farm, per clustered distance ((0, 3], (3, 6] . . . (45, 48]), and one without distinction (Fig. 3).

The final model was a linear ensemble of the previously mentioned models (nine models). To define the ensemble weights, a simple linear regression optimizing the RMSE (Eq. (6)) was used. It was trained using the cross-validated predictions of the models. It also used *farm* and *dist* as features, because they were used to do the splitting, and had a relationship with each model's weaknesses and strengths

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(X_{\mathrm{obs},i} - X_{\mathrm{model},i})^2}{n}}. \quad (6)$$

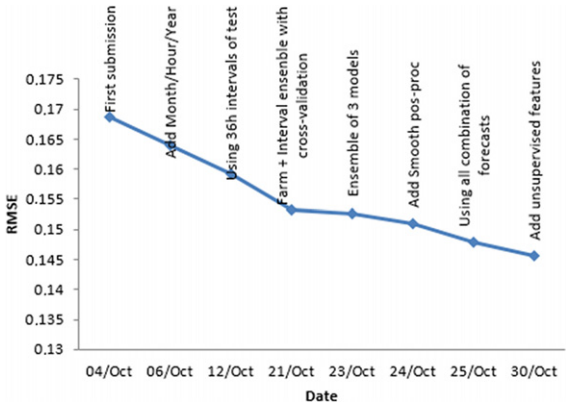

Fig. 7. Score evolution.

Table 7
Final standings.

Rank	Team	Public score	Private score
1	Leustagos	0.14574	0.14567
2	DuckTile	0.14872	0.14720
3	Gilberto Titericz Jr	0.14792	0.14822
4	Stefan Henß	0.14684	0.14854
5	Mz	0.14804	0.14916

After that, yet another model was trained, this time a smoothing one. It took the result of the ensemble and did a $T \pm n$ ($n = -3$ to 3) interpolation of the predictions. This model became almost useless when $ws.angle(T \pm n)$ was included in the previous ones, but it did improve the prediction a bit, which was useful for the competition.

3. Results and discussion

3.1. Results

The evaluation metric was the RMSE (Eq. (6)), the root mean squared error. The value of this metric goes down as the accuracy of the model goes up. Fig. 7 shows the score improvement of the milestones in this solution. As can be seen, the first model, containing only forecasts and previous known values, scored 0.1685 RMSE. Then, by adding some seasonal features (hour, month and year), it decreased to 0.16393. Next were included in the set the intervals of 36 h that were available in the test period, and it improved again. After the cross-validation scheme was changed so as to do the validation splitting using the whole 36 + 48, the score jumped to 0.15315. Next, adding the overall model and the smoothing model of final predictions took it to 0.15103. The last big leap in the score was the inclusion of all combinations of dates and forecasts, with four forecasts for each known instance, which scored 0.14779. Finally, the unsupervised cluster features were included, giving the final and best score of 0.14567.

Looking at the final results in Table 7, we can see that the whole procedure generalizes well, generating consistent results in both the public and private leaderboards.

3.2. Discussion

Here, we have provided an efficient approach to the prediction of wind power generation on seven distinct wind farms. The solution was made up of two parts: first, extracting features, and second, applying distinct aggregation schemes to the data, in order to create models. In the end, this approach used many models, more than would be needed in an actual situation, but only because this approach was followed in order to win a competition, when every bit of performance must be considered, disregarding the computation expenses required to build it. Also, some data that would be known in real operations were hidden, increasing the complexity of the training process. After the competition, some tests were performed with the models and a simpler version of it the process was obtained, a version which performed almost as well, but was much less complicated. In that approach, only the two overall GBM regressions were used, and the features which were built to merge the wind strength and direction were discarded. It might be possible to achieve an even better accuracy by including more features, like temperature and humidity forecasts.

4. Conclusion

We have shown that with some clever combining of well-known algorithms (GBM, linear regression and K-Means) it is possible to achieve high levels of precision for wind power prediction. The most important steps were choosing a reliable scheme of validation for training and creating good features. The key for the result was never to forget the time series nature of the dataset. Many of the features, some post-processing steps and the validation scheme used this characteristic in their design, and were the source of this approach's success. The proposed solution was somewhat complex but accurate. Some tests also showed that it could be simplified greatly, with only a marginal performance degradation.

References

Friedman, J. H. (2001). *Greedy function approximation: a gradient boosting machine*. Reitz lecture.
Geyer, C. J. (2003). Generalized linear models in R. http://www.stat.umn.edu/geyer/5931/mle/glm.pdf.
Hong, T., Pinson, P., & Fan, S. (2013). Global energy forecasting competition 2012. In *GEFCom 2013*.
Ridgeway, G. (2013). gbm: generalized boosted regression models. arxiv:cran.r-project.org/package=gbm.