# On Heterogeneous Machine Learning Ensembles for Wind Power Prediction

**Justin Heinermann and Oliver Kramer**
Department of Computing Science, University of Oldenburg, 26111 Oldenburg, Germany
{justin.philipp.heinermann, oliver.kramer}@uni-oldenburg.de

**Abstract**

For a sustainable integration of wind power into the electricity grid, a precise prediction method is required. In this work, we investigate the use of heterogeneous machine learning ensembles for wind power prediction. We first analyze homogeneous ensemble regressors that make use of a single base algorithm and compare decision trees to k-nearest neighbors and support vector regression. As next step, we construct heterogeneous ensembles that make use of multiple base algorithms and benefit from a gain of diversity of the weak predictors. In the experimental evaluation, we show that a combination of decision trees and support vector regression outperforms state-of-the-art predictors (improvements of up to $37\%$ compared to support vector regression) as well as homogeneous ensembles while requiring a shorter runtime (speed-ups from $1.60\times$ to $8.78\times$). The experiments are based on large wind time series data from simulations and real measurements.

## 1 Introduction

Precise predictions of the wind power production are crucial for the integration of wind turbines into the power grid. Besides numerical weather predictions, machine learning algorithms yield good forecasting results (Kramer, Treiber, and Gieseke 2013; Salcedo-Sanz et al. 2014). E.g., support vector regression (SVR) or $k$-nearest neighbors ($k$-NN) regression can provide suitable results for short-term forecast horizons. However, there are two main drawbacks: First, in order to reach the best prediction accuracy possible with these algorithms, the computation time for both training and testing grows infeasibly large. Second, the prediction performance needs to be improved further to cope with the actual energy markets needs.

It has been shown that a good alternative to the well-known machine learning algorithms is to combine several basic models to machine learning ensembles: By employing a number of so-called weak predictors, which use standard machine learning algorithms, and combining their outputs, the accuracy of classification and regression can be ameliorated while reducing the computation time, see Dietterich (2000). Especially for real-world problems, machine learning ensembles are promising approaches. In contrast to state-of-the-art machine learning algorithms, ensemble methods require less tuning and expert domain knowledge. Nevertheless, in order to find an optimal ensemble predictor, usually a trade-off between multiple objectives has to be made: For example, a lower prediction error is often achieved by investing more computation time, see Rokach (2010).

In this work, we discuss the practical use of bagging ensembles for the task of wind power forecasting, which can be formulated as regression problem. We investigate which ensemble setting can offer the best results in a reasonable runtime. In the first step, we compare homogeneous ensemble predictors, where every weak predictor makes use of the same base algorithm. In this paper, we use decision trees, $k$-NN regression, and support vector regression as base algorithms.

Going further, we propose the use of heterogeneous ensemble predictors for wind power prediction. We investigate the question, if heterogeneous ensembles that consist of different types of base predictors, perform better than homogeneous ensembles. A reason for the success is the gain of diversity of the weak predictors. Our comprehensive experimental results show that a combination of Decision Trees (DT) and SVR yields better results than the analyzed homogeneous predictors while offering a decent runtime behavior.

### 1.1 Related Work

Kusiak, Zhang and Song (2009) successfully apply different methods to short-term wind power prediction, one of which is the bagging trees algorithm. Fugon *et al.* (2008) compare various algorithms for wind power forecasting and show that random forests with and without random input selection yield a prediction performance similar to SVR, but recommend to prefer a linear model when the computation time grows too large. Heinermann and Kramer (2014) achieve good wind power prediction results using support vector regression ensembles.

In the field of numerical weather forecasts, it is quite common to use ensemble postprocessing: Gneiting *et al.* (2005) found ensembles to reduce the prediction error by applying the ensemble model output statistics (EMOS) method to diverse weather forecasts. Similarly, Thorarinsdottir and Gneiting (2010) are using a so-called "heteroscedastic cen-

sored regression" for maximum wind speed prediction over the American Pacific Northwest.

A similar domain to wind power prediction is time series prediction for solar power output: Chakraborty *et al.* (2012) built up an ensemble of a weather forecast-driven Naïve Bayes Predictor as well as a $k$NN-based and a Motif-based machine learning predictor. The results of the three predictors are combined with a Bayesian Model Averaging. Chakraborty *et al.* show that the prediction error can be reduced by inducing ensemble methods to forecasting power output.

## 2 Background

In this section, we briefly describe our machine learning approach to wind power prediction and provide the relevant background to machine learning ensembles.

### 2.1 Machine Learning Approach to Wind Power Prediction

Short term wind power prediction can be treated as a regression problem. In contrast to numerical weather predictions, statistical learning methods usually make only use of the time series data itself. The objective is to predict the measurement after a *forecast horizon* $\lambda$. The input patterns consist of a past of $\mu$ time steps, which we call the *feature window*. In this work, we use a spatio-temporal model based on the one proposed by Kramer, Treiber, and Gieseke (2013), which showed the benefit of involving neighboring turbines to the input vector. Let $p_i(t)$ be the measurement of a turbine $i$ at a time $t$, and $1 \leq i \leq m$ the indices of the $m$ neighboring turbines. Then, for a target turbine with index $j$ we define a pattern-label-pair $(\mathbf{x}, y)$ for a given time $t_0$ as

$$\begin{pmatrix} p_1(t_0-\mu) & \dots & p_1(t_0) \\ \dots & \dots & \dots \\ p_m(t_0-\mu) & \dots & p_m(t_0) \end{pmatrix} \rightarrow p_j(t_0+\lambda) \qquad (1)$$

In our experiments, we use the *NREL western wind resources dataset*[1]. It consists of simulated wind power output for $32,043$ wind turbines in the US, given in 10-minute time resolution for the years 2004 - 2006. For every turbine, there are $157,680$ wind speed and power output measurements available.

### 2.2 Machine Learning Ensembles for Supervised Learning

The idea of ensemble methods can be described as as building "a predictive model by integrating multiple models" (Rokach 2010). One of the advantages is the possible improvement of prediction performance. Another reason for utilizing ensemble methods is the reduction of computational complexity, which can be helpful on very large data sets. There are countless variants of different ensemble algorithms. A comprehensive overview and empirical analysis for ensemble classification is given by Bauer and Kohavi (1999). Another, more up-to-date review paper is given by Rokach (2010).

[1] http://wind.nrel.gov/

An important and famous approach is *bagging*, which stands for bootstrap aggregating, and was introduced by Breiman (1996). The main idea is to build independent predictors using samples of the training set and average (or vote) the output of these predictors. Breiman shows that bagging ensembles of decision trees as well as regression trees work very well in comparison with single trees. Furthermore, he gives arguments for the question "why bagging works" (Breiman 1996). A popular variant of bagging approaches is the random forest algorithm (Breiman 2001) that "uses a large number of individual, unpruned decision trees" (Rokach 2010). Every decision tree is built with a subset sample from the training set, but only uses $N$ of the available features of the patterns.

There exist more sophisticated ensemble approaches like AdaBoost (Freund, Schapire, and others 1996) or *Stacked Generalization* (Wolpert 1992), but, as we want to give a proof of concept, we limit ourselves here to bagging.

One key ingredient to successful building of ensembles is the concept of diversity: All the weak predictors should behave different if not uncorrelated (Rokach 2010; Brown et al. 2005). Then, the ensemble prediction improves. There are many ways to generate such diversity, like manipulating the used training sample, the used features, and the weak predictors' parameters. Another possibility is the hybridization. An example is given by Kramer *et al.* (2012): support vector machine and $k$-NN classification are combined and complement each others behavior.

## 3 Regression Ensembles for Wind Power Prediction

Our objective is to find out, if heterogeneous machine learning ensembles are a superior alternative to state-of-the-art machine learning predictors. We decided to implement a relatively simple bagging approach with weighting, which has some advantages. While the implementation is straightforward and offers a moderate computational cost, we consider the approach sufficient for a proof of concept, which is also shown in the experimental evaluation. Another motivation for this kind of ensemble algorithm is that the famous Random Forest method yields very good results, too, and is relatively fast compared to boosting algorithms. The latter ones could outperform bagging algorithms for some applications, but also tend to overfitting, and can hardly be parallelized. A comparison to more sophisticated ensemble approaches like AdaBoost as well as stacked generalization will be subject to future work.

### 3.1 Ensemble Algorithm

Our algorithm is outlined in Algorithm 1. As usual in supervised learning, a training set $X_{train}$ with known labels is given. The most important meta-parameters of the algorithm are the number $n$ of weak predictors, the number $s$ of samples, and the types of base algorithms used for each predictor. Both $n$ and $s$ have to be chosen large enough in order to provide satisfying results. However, the best choice for $n$ and $s$ depends on the base algorithm used, which also influences the runtime significantly. The main work of the algo-

**Algorithm 1** Training of Ensemble Predictor

1: **Inputs:**
   $X_{train} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_m, y_m)\} \subset \mathbb{R}^d \times \mathbb{R}$
   Number of predictors: $n$
   Number of samples: $s$
   Types of Algorithm to use: A=$\{a_i | i \in 1 \ldots n\}$
2: **Returns:**
   Predictors: $P = \{p_i | i = 1, \ldots, n\}$
   Weights: $W = \{w_i \in \mathbb{R} | i = 1, \ldots, n\}$
3: **for** i = 1 **to** $n$ **do**
4: $X_{sample} \leftarrow sample(X_{train}, s)$
5: $X_{val} \leftarrow X_{train} - X_{sample}$
6: $p_i \leftarrow trainPredictor(a_i, X_{sample})$
7: $w_i \leftarrow \frac{1}{MSE(p_i, X_{val})}$
8: **end for**

rithm takes place in the `for`-loop beginning in line 5. Each pass trains one weak predictor $p_i$ and its assigned weight $w_i$. For each weak predictor, a subset of $X_{train}$ with size $s$ is sampled and used as training set $T_i$ for the particular predictor $p_i$. In order to calculate weight $w_i$, the remaining training patterns are used as a validation set $T_{val}$. The value $w_i$ is then obtained by testing $p_i$ on $T_{val}$ and taking the inverse of the mean squared error (MSE).

When the training algorithm computed the predictors and weights, for an unknown distance $\mathbf{x}$ the predicted label is computed by:

$$\widehat{y} = \frac{\sum_{i=1}^{k} w_i \cdot p_i(\mathbf{x})}{\sum_{i=1}^{k} w_i} \tag{2}$$

Each predictors output $p_i(\mathbf{x})$ is computed and then weighted by $w_i$ in the resulting weighted average. In a realistic scenario, one would perform all calls of $p_i$ in parallel using multi- or manycore processors. In our experiments, we only employed one CPU core for the runtime measurements.

As pointed out by Ho (1998), random forests and bagging classifiers in general my benefit from sampling from the feature space, i.e., taking only a random subset of the available features into account. Besides the number of features used, the choice can be done with or without replacement. Although replacement is sometimes considered as useful (Rokach 2010; Dietterich 2000), there is no explicit rule when to use it. In a preliminary experiment, we found no evidence that random feature subspaces help to increase the accuracy, but of course can help to decrease runtime. Because we did not find a significant difference between sampling with replacement and sampling without replacement, we employ sampling without replacement. For the basic comparison of the weak predictors, all features are used. For the comparison of heterogeneous ensembles with state-of-the-art predictors, the number of features sampled will be considered again because of the possible trade-off between runtime and accuracy. Concerning the sampling from the training set, we also found no evidence for the supremacy of replacement, but due to the recommendations in literature (Breiman 1996; Rokach 2010), we decided to employ sampling with replacement in the following experiments.

### 3.2 Choice of the Base Algorithms and Training of Weak Predictors

Since the number of possible settings is huge, one has to make some assumptions to limit the number of combinations. First, we want to give an overview over the choice of base algorithms and parameters.

The decision tree algorithm is a powerful yet simple tool for supervised learning problems (Hastie, Tibshirani, and Friedman 2009). The main idea is to build a binary tree, which partitions the feature space into a set of rectangles. The assignment of the unknown pattern to one of these rectangles is used for computing the sought label. While there are different algorithms for building up decision trees, we limit ourselves to the famous CART algorithm (Breiman et al. 1984). Besides moderate computational costs, the main advantage of decision trees is their interpretability.

The SVR algorithm often provides very good prediction results and is regarded as state-of-the-art regression technique. In general, the support vector machine (SVM) algorithm maximizes a geometric margin between the instances of the classes to be separated. Similar to SVM classification, the SVR algorithm aims at finding a prediction function $\hat{f} : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}$ that computes an accurate prediction value for an unseen pattern $\mathbf{x} \in \mathbb{R}^d$. For more detailed information about the algorithm, we refer to, e.g., (Steinwart and Christmann 2008). We utilized a RBF kernel and choose regularization parameter $C = 10,000$ and kernel bandwidth $\sigma = 1e-5$.

A famous yet relatively simple approach for classification and regression is the *$k$-nearest neighbors* ($k$-NN) model, see (Hastie, Tibshirani, and Friedman 2009). The prediction averages the label information of the $k$ nearest neighbors, i.e., via $f(\mathbf{x}) = \frac{1}{k} \sum_{i \in N_k(\mathbf{x})} y_i$, where $N_k$ denotes the set of indices for the $k$ nearest neighbors in $T$ w.r.t. a distance metric like Euclidean distance. While a naïve implementation takes $\mathcal{O}(|S| \cdot |T|)$ time for a training set $T$ and a test set $S$, more efficient implementations with spatial data structures, e.g. $k$-d trees, are available (Hastie, Tibshirani, and Friedman 2009). Logarithmic runtime is offered for small dimensionalities $d \leq 15$. For parameter $k$, we make a random choice in the interval $[1; 25]$.

In a preliminary experiment, we compare different regression algorithms composed to ensembles: Table 1 shows a comparison of decision trees, SVR, and $k$-NN used as weak predictors. A general observation is that increasing $n$ and $s$ decreases the prediction error. With the given $n$ and $s$, no clear decision between decision trees and SVR can be made, but we stick to these two basic algorithms in the further experiments rather than $k$-NN.

### 3.3 Heterogeneous Ensembles

Of course, when dealing with forecasting tasks, the first goal is to reach the lowest error possible. While it is possible to decrease the prediction error by using ensembles, a feasible runtime is equally important. If it takes hours to train a regressor for one turbine, a model would be unusable for a large number of turbines to be forecasted – the time needed for parameter-tuning and cross-validation not

Table 1: Comparison (MSE) of ensemble predictors consisting of different base algorithms used as weak predictor. For every turbine, the best result is printed in bold. Each experiment has been repeated 10 times.

| Base Algorithm | Decision Tree | | | | SVR | | | | $k$-NN | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 32 | 32 | 256 | 256 | 32 | 32 | 256 | 256 | 32 | 32 | 256 | 256 |
| $s$ | 500 | 1,000 | 500 | 1,000 | 500 | 1,000 | 500 | 1,000 | 500 | 1,000 | 500 | 1,000 |
| Casper | 11.17 | 10.93 | 10.89 | **10.62** | 10.99 | 10.84 | 10.95 | 10.87 | 13.10 | 12.44 | 13.02 | 12.44 |
| Las Vegas | 10.84 | 10.61 | 10.51 | 10.27 | **10.26** | **10.26** | 10.27 | 10.27 | 12.84 | 12.36 | 12.81 | 12.30 |
| Hesperia | 7.98 | 7.82 | 7.76 | **7.59** | 7.62 | 7.61 | 7.60 | **7.59** | 9.41 | 8.96 | 9.36 | 8.98 |
| Reno | 14.76 | 14.53 | 14.47 | 14.19 | 14.11 | 14.10 | 14.00 | **13.98** | 18.92 | 18.03 | 19.14 | 18.14 |
| Vantage | 7.31 | 6.97 | 7.00 | 6.83 | 6.61 | 6.58 | **6.57** | **6.57** | 8.44 | 7.93 | 8.43 | 8.07 |

mentioned. Therefore, our goal is to reach a good prediction performance as well as a short runtime for both training and testing.

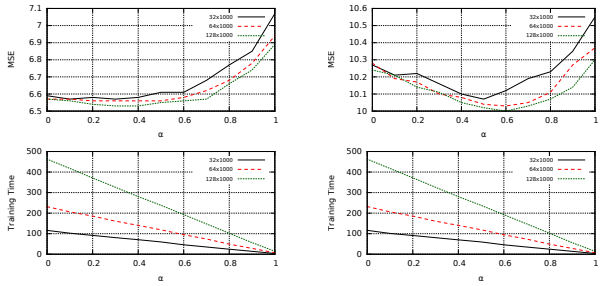
As seen in Table 1, there is no clear answer which algorithm to prefer. We propose to use heterogeneous ensembles built upon SVR and decision tree regressors. As shown in the following experiments, the result is a robust prediction algorithm that offers a reasonable runtime behavior. The experiments in the following section address the question, if heterogeneous ensembles offer a better performance than homogeneous ensembles. The second question is: Can heterogeneous ensembles help to decrease the computation time needed?

## 4 Experimental Results

In our experiments, we analyze heterogeneous ensembles built upon SVR and decision tree regressors. As data source, we use the *NREL western wind resources dataset*. Each grid point has a maximum power output of $30MW$ and 10-minute data is given for the years 2004 - 2006. Therefore, for every station there are $157,680$ wind speed and power output measurements available. In our experiments, we use the power output data of ten wind parks[2] that consist of the target wind turbine and the 10 neighbored turbines. Therefore, As training data set, the data from $01/2004$ until $06/2005$ is used and the data from $7/2005$ until $12/2006$ serves as test data set. The experiments were run on an Intel Core i5 at $3.10GHz$ with $8GB$ of RAM. The algorithms were implemented in Python utilizing the $k$NN, decision tree, and SVR implementations of *Scikit-learn* (Pedregosa et al. 2011). As metric for the prediction performance we use the MSE.

### 4.1 Mixed Ensemble with Coefficient $\alpha$

First, we analyze heterogeneous ensembles that employ both SVR predictors and decision trees. We define a coefficient $\alpha$ that specifies the amount of weak predictors trained with the decision tree algorithm. Hence, $1-\alpha$ determines the number of SVR predictors in the ensemble. Figure 1 shows the experimental results for two wind turbines. For both (a) and (b), three ensemble settings were analyzed showing MSE and training time: $s$ is set to $1,000$ and $n$ is set to 32, 64, or 128. The higher the number of predictors is, the lower the prediction error becomes. But also the runtime increase is notably.[3] With $\alpha = 0$, we observe an ensemble with only SVR predictors, with $\alpha = 1$, only decision trees are chosen. The interesting point is that, given a sufficient number $n$, the prediction quality becomes maximal with an $\alpha$ in the middle range. This points to an advantageous behavior of heterogeneous ensembles: With an $\alpha$ of approximately $0.5$, the training and testing time of the ensemble predictor decreases dramatically compared to $\alpha = 0.0$. Therefore, the parameter $\alpha$ can be seen as explicit tuning parameter for the trade-off between prediction performance and runtime.




(a) Mixture $\alpha$ for a wind turbine near Vantage

(b) Mixture $\alpha$ for a wind turbine near Las Vegas

Figure 1: Mixing of SVR and DT. With a balanced mixture, a better MSE is reached within a shorter training time.

### 4.2 Meta-Ensemble Combining SVR Ensembles and Decision Tree Ensembles

While the results of the former experiment are promising, we have to point out that the parameter $\alpha$ was varied for fixed $n$ and $s$. One has to consider that different weak predictors show different behavior and could benefit from different combinations of $n$ and $s$. I.e., we will see that a large amount of predictors is a good possibility to ameliorate decision tree ensembles while the runtime does not suffer as much as in SVR ensembles. Instead of combining some SVR predictors and some decision trees, one could possibly better combine a huge number of decision trees and maybe also increase sample number $s$.

Therefore, we have to give a fair comparison, which considers both prediction performance and runtime. First, we

[2]The corresponding IDs of the turbines in the NREL dataset are: Casper = 23167, Hesperia = 2028, Las Vegas = 6272, Reno = 11637, Vantage = 28981

[3]The testing time was similar to the training time.

Table 2: Behavior of different setups of ensembles, which are then combined.

(a) SVR ensembles

| setup | $n$ | $s$ | $t_{train}$ | $t_{test}$ | MSE |
|---|---|---|---|---|---|
| 1 | 4 | 500 | 7.58 | 7.11 | 10.78 |
| 2 | 4 | 1,000 | 14.69 | 13.09 | 10.43 |
| 3 | 4 | 2,000 | 29.94 | 25.46 | 10.58 |
| 4 | 32 | 500 | 59.73 | 55.91 | 10.30 |
| 5 | 32 | 1,000 | 117.86 | 105.61 | 10.26 |
| 6 | 32 | 2,000 | 245.85 | 206.68 | 10.35 |
| 7 | 64 | 500 | 119.40 | 111.91 | 10.20 |
| 8 | 64 | 1,000 | 236.26 | 211.83 | 10.26 |
| 9 | 64 | 2,000 | 489.92 | 410.81 | 10.26 |

(b) Decision Tree Ensembles

| setup | $n$ | $s$ | $t_{train}$ | $t_{test}$ | MSE |
|---|---|---|---|---|---|
| 1 | 32 | 2000 | 4.91 | 3.65 | 10.44 |
| 2 | 32 | 4,000 | 7.74 | 3.85 | 10.27 |
| 3 | 32 | 40,000 | 81.96 | 3.82 | 10.06 |
| 4 | 256 | 2,000 | 42.67 | 28.86 | 10.13 |
| 5 | 256 | 4,000 | 63.17 | 29.17 | 10.05 |
| 6 | 256 | 40,000 | 733.15 | 30.35 | 9.86 |
| 7 | 1,024 | 2,000 | 161.60 | 113.97 | 10.15 |
| 8 | 1,024 | 4,000 | 258.03 | 115.55 | 10.03 |
| 9 | 1,024 | 40,000 | 2,859.18 | 136.71 | 9.82 |

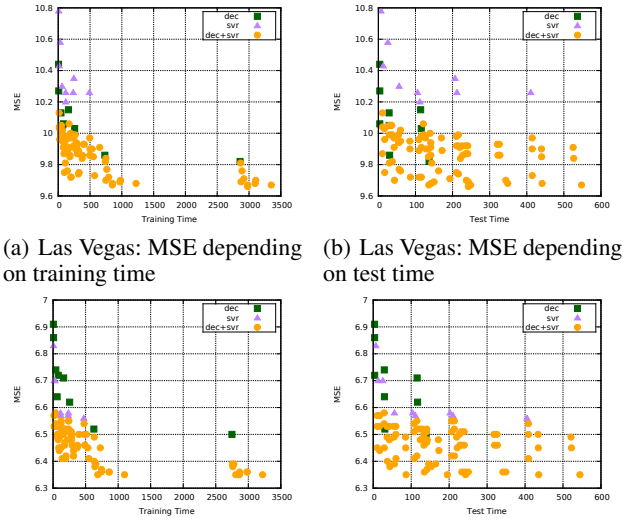
(c) Combinations of one DT ensemble and on SVR ensemble to one heterogeneous ensemble. For every combination, the MSE is shown. The row denotes which SVR ensemble is employed, whereas the column shows which DT ensemble is used.

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 |
|---|---|---|---|---|---|---|---|---|---|
| S1 | 10.13 | 10.04 | 9.87 | 10.05 | 9.99 | 9.82 | 10.06 | 9.99 | 9.81 |
| S2 | 10.03 | 9.96 | 9.75 | 9.97 | 9.91 | 9.70 | 9.97 | 9.90 | 9.69 |
| S3 | 10.05 | 9.99 | 9.81 | 9.99 | 9.94 | 9.77 | 10.00 | 9.94 | 9.76 |
| S4 | 10.02 | 9.95 | 9.76 | 9.95 | 9.90 | 9.72 | 9.97 | 9.89 | 9.71 |
| S5 | 9.96 | 9.89 | 9.72 | 9.90 | 9.85 | 9.67 | 9.91 | 9.84 | 9.66 |
| S6 | 9.98 | 9.92 | 9.75 | 9.92 | 9.87 | 9.70 | 9.93 | 9.86 | 9.70 |
| S7 | 9.98 | 9.91 | 9.72 | 9.91 | 9.86 | 9.68 | 9.92 | 9.86 | 9.67 |
| S8 | 9.99 | 9.92 | 9.74 | 9.92 | 9.87 | 9.69 | 9.93 | 9.86 | 9.68 |
| S9 | 9.97 | 9.90 | 9.73 | 9.90 | 9.85 | 9.68 | 9.91 | 9.84 | 9.67 |

analyze the behavior of the homogeneous ensembles based on SVR or decision trees. We try to find good combinations, which are computable in a feasible time. The result can be seen in Table 2: Like assumed, one can train more decision trees with a larger sample in the same time than SVR predictors. The central point of the experiment is the equally-weighted combination of one SVR ensemble and one DT ensemble at a time to one heterogeneous ensemble. The results of these combinations are depicted in Table 2(c), which has the form of a matrix. In every cell of the matrix, the used SVR ensemble is given by the row and the used decision tree ensemble is given by the column. In the table, only the MSE is given for clear arrangement. The training and test times for one predictor is approximately the sum of the respective times of the two combined ensembles. Besides the very promising results, which outperform the homogeneous ensembles, we also can see that the combination of two weaker ensembles takes less time to deliver the same prediction error. We visualize this behavior for two wind turbines in Figure 2.

### 4.3 Comparison with State-of-the-Art Predictors

At last, we give a comparison of our heterogeneous ensemble method to SVR, which is considered as state-of-the-art regressor. Because SVR outperformed $k$-NN in all cases, we do not visualize $k$-NN results. The parameters $k$ and accordingly $C$ and $\sigma$ were optimized with a 10-fold cross-validation. The training times are measured using the optimal parameters, so the huge amount for parameter tuning is not included. The testing times showed similar behavior. In Figure 3, we visualized three setups: First, standard SVR with RBF kernel and tuned $C$ and $\sigma$. As comparison, we show two heterogeneous ensembles in the style of Section 4.2: Both consist of one SVR ensemble with $n = 32$ an $s = 1,000$ and one decision tree ensemble with $n = 256$ and $s = 10,000$. The first one uses all 33 features available, whereas the second only makes use of 15 randomly chosen features without replacement. The comparison shows that in most cases the ensemble predictor outperforms classical SVR. Further, the training time is much shorter. If one must make a trade-off and decrease training or testing time, he might want to use a feature-reduced variant.




(a) Las Vegas: MSE depending on training time

(b) Las Vegas: MSE depending on test time

(c) Vantage: MSE depending on training time

(d) Vantage: MSE depending on test time

Figure 2: Behavior of runtime and prediction performance for homogeneous and heterogeneous ensembles for two wind turbines near Las Vegas and Vantage. The heterogeneous combinations can outperform the homogeneous ensembles. In particular, the solutions in each bottom left corner show ensembles with a very short computation time as well as a very low error.

## 5 Conclusions

Wind power can only be integrated into the power grid, if a reliable forecast can be given in a reasonable time. After analyzing different types of ensemble predictors, we presented a heterogeneous ensemble approach utilizing both DT and SVR. In our comprehensive experimental evaluation, we showed that our approach yields better results within a much shorter computation time. The trade-off between prediction performance and computation time can easily be managed by adapting the parameters like number of predictors, number of samples, and number of features used. In the future, we plan to implement automatic ensemble optimization and the integration of other regression algorithms.

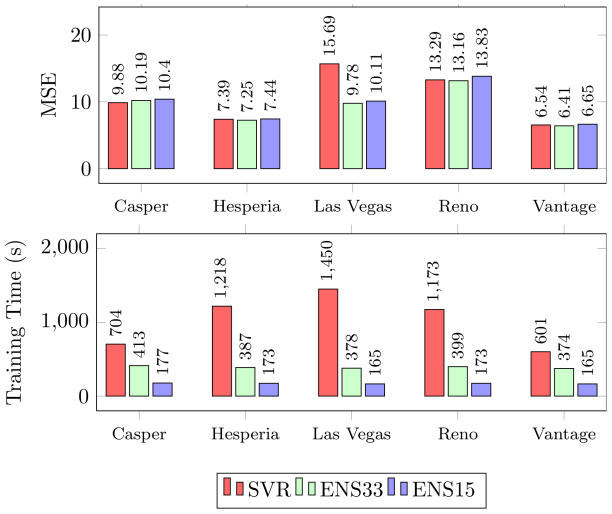



Figure 3: Comparison of MSE and training time for five wind turbines. Our ensemble using all features yields the best MSE in four cases, but only takes a small amount of the time taken by standard SVR. If training time is considered more important than MSE, on can reduce the number of features used without loosing much of prediction performance.

## Acknowledgements

We thank the ministry of science and culture of Lower Saxony for supporting us with the PhD Programm *System Integration of Renewable Energies*. Furthermore, we thank the US *National Renewable Energy Laboratory* (NREL) for providing the wind data set.

## References

Bauer, E., and Kohavi, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning* 36(1-2):105–139.

Breiman, L.; Friedman, J.; Stone, C.; and Olshen, R. 1984. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Breiman, L. 1996. Bagging predictors. *Machine learning* 24(2):123–140.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Brown, G.; Wyatt, J.; Harris, R.; and Yao, X. 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* 6(1):5–20.

Chakraborty, P.; Marwah, M.; Arlitt, M. F.; and Ramakrishnan, N. 2012. Fine-Grained Photovoltaic Output Prediction Using a Bayesian Ensemble. In *AAAI Conference on Artificial Intelligence*.

Dietterich, T. 2000. *Ensemble Methods in Machine Learning*, volume 1857 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 1–15.

Freund, Y.; Schapire, R. E.; et al. 1996. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, volume 96, 148–156.

Fugon, L.; Juban, J.; Kariniotakis, G.; et al. 2008. Data mining for wind power forecasting. In *Proceedings European Wind Energy Conference & Exhibition EWEC 2008*.

Gneiting, T.; Raftery, A. E.; Westveld, A. H.; and Goldman, T. 2005. Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation. *Monthly Weather Review* 133(5):1098–1118.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning*. Springer, 2nd edition.

Heinermann, J., and Kramer, O. 2014. Precise Wind Power Prediction with SVM Ensemble Regression. In *Artificial Neural Networks and Machine Learning–ICANN 2014*, 797–804. Springer.

Ho, T. K. 1998. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20(8):832–844.

Kramer, O.; Wilken, O.; Beenken, P.; Hein, A.; Hüwel, A.; Klingenberg, T.; Meinecke, C.; Raabe, T.; and Sonnenschein, M. 2012. On ensemble classifiers for nonintrusive appliance load monitoring. In *Hybrid Artificial Intelligent Systems*. Springer. 322–331.

Kramer, O.; Treiber, N. A.; and Gieseke, F. 2013. Machine Learning in Wind Energy Information Systems. In *EnviroInfo*, 16–24.

Kusiak, A.; Zheng, H.; and Song, Z. 2009. Short-term prediction of wind farm power: a data mining approach. *Energy Conversion, IEEE Transactions on* 24(1):125–136.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Rokach, L. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33(1-2):1–39.

Salcedo-Sanz, S.; Rojo-Álvarez, J.; Martínez-Ramón, M.; and Camps-Valls, G. 2014. Support vector machines in engineering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4(3):234–267.

Steinwart, I., and Christmann, A. 2008. *Support Vector Machines*. Information science and statistics. Springer.

Thorarinsdottir, T. L., and Gneiting, T. 2010. Probabilistic forecasts of wind speed: ensemble model output statistics by using heteroscedastic censored regression. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 173(2):371–388.

Wolpert, D. H. 1992. Stacked generalization. *Neural networks* 5(2):241–259.