Heterogeneous Ensembles for Short-Term Electricity Demand Forecasting

Grzegorz Dudek
Department of Electrical Engineering
Czestochowa University of Technology
42-200 Czestochowa, Al. Armii Krajowej 17, Poland
dudek@el.pcz.czest.pl

***Abstract*—In this work multi-model ensembles are proposed for short-term electricity demand forecasting. The ensembles are composed of ten members representing different model classes. The base models are integrated using simple averaging or dynamically weighted averaging, where weights depend on the model performance on the forecasting tasks similar to the current one. Simulation studies on different load time series show that the ensemble errors are lower than the mean errors for the base models, and in most cases even lower than error of the best base model. The variance of the forecasts is also reduced.**

***Keywords*—ensemble forecasting; multi-model ensemble; pattern-based forecasting; short-term load forecasting**

I. Introduction

In recent years ensemble learning systems have been widely used in machine learning, computational intelligence and data mining. They are composed of many base learners or models, each of which provides an estimate of a target function. The learner responds are combined in some way to produce a common response, which has a smaller generalization error compared to a single learner.

In forecasting the target is usually continuous and an ensemble goal is to find a useful approximation to the target function that underlies the predictive relationship between the predictors and forecasted variable. The multiple estimates of the target function generated by the ensemble members are integrated usually by a linear combination. The weights assigned to members in the linear combination can be the same for each member or can be dependent on the individual learner performance.

The ensemble learning process can be divided into three steps [1]:

1. ensemble generation, in which a set of redundant models is generated,
2. ensemble pruning, in which some of the models are removed,
3. ensemble integration, in which the models are combined.

In many applications of ensembles the second step, in which we try to prune a trained ensemble to get a better performance, is often omitted. In ensemble learning a fundamental issue is ensemble diversity, i.e. the difference among the individual learners. A good tradeoff between performance and diversity of members underlies the success of ensemble learning. The more accurate and the more diverse the individual learners, the better the ensemble. It was shown in [2] that the error of the ensemble will never be higher than the average error of the individual learners. Generating diverse learners is a challenging problem. Diversity in members is achieved by learning them on different subsets of the training set (e.g. bootstrap sampling in bagging), on different input features (random subspace methods), using different model architectures, different learning algorithms, different starting points, different learning parameters or different output representations [3]. Sometimes several of these approaches are used in the same time.

The component models can be of the same type or of different types. In the first case the ensemble is called homogeneous, in the second case it is called heterogeneous. Different nature of the base learners are also the source of diversity in heterogeneous ensembles. Some experimental results show that heterogeneous ensembles can improve accuracy compared to homogenous ones [4]. This is because the error terms of models of different types are less correlated than the errors of models of the same type.

In this work a heterogeneous ensemble is proposed for short-term electricity demand forecasting. Ensemble members are strong learners trained independently on the same data. They are described in Section II. Two ways of integration of the forecasts generated by the base models are considered: simple averaging and linear combination with weights dynamically adjusted according to the model performance. This is presented in Section III. In Section IV the proposed approach is evaluated over several real world datasets.

II. Ensemble Members

The proposed ensemble comprises ten forecasting models. They are based on: Nadaraya-Watson estimator, fuzzy neighborhood model, *k*-means clustering (two models), artificial immune systems (tree models), ARIMA, exponential smoothing and neural networks. All models except ARIMA and exponential smoothing work on patterns of the daily cycles of the load time series. An input pattern $\mathbf{x}_i = [x_{i,1}\ x_{i,2}\ \ldots\ x_{i,n}]$ are normalized load vector $\mathbf{L}_i = [L_{i,1}\ L_{i,2}\ \ldots\ L_{i,n}]$

representing daily curve preceding the forecasted day. Its components are defined as follows [5]:

$$x_{i,t} = \frac{L_{i,t} - \overline{L}_i}{\sqrt{\sum_{l=1}^{n}(L_{i,l} - \overline{L}_i)^2}}, \tag{1}$$

where $L_{i,t}$ is the power system load in period t of the day i, $\overline{L}_i$ is the mean load of the day i and n is the number of periods t in daily period, typically 24, 48 or 96.

An output or forecast pattern $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,n}]$ maps the load vector $\mathbf{L}_{i+\tau} = [L_{i+\tau,1}\ L_{i+\tau,2}\ \ldots\ L_{i+\tau,n}]$ representing the forecasted daily curve. The components of $\mathbf{y}_i$ are defined as follows:

$$y_{i,t} = \frac{L_{i+\tau,t} - \overline{L}_i}{\sqrt{\sum_{l=1}^{n}(L_{i,l} - \overline{L}_i)^2}}, \tag{2}$$

where $\tau = 1, 2, \ldots$ is a forecast horizon in days.

The goal of daily curves preprocessing using patterns is their unification. The x- and y-patterns expresses unified shapes of daily curves. Annual and weekly cycles and also trend are filtered out. Consequently, the relationship between predictors and forecasted variable simplifies and is easier to capture by the model.

For the given horizon τ the forecasting models learn the relationship $\mathbf{x} \rightarrow \mathbf{y}$ on the training set $\Omega = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$. The forecasted load vector $\mathbf{L}_{i+\tau}$ is calculated from transformed equation (2) after the forecast of pattern $\mathbf{y}$ is generated by the model.

A. Nadaraya-Watson Estimator

The pattern-based forecasting model using Nadaraya-Watson estimator (N-WE) was described in [6]. This is a nonparametric multivariate regression model of the form:

$$h^1(\mathbf{x}) = \frac{\sum_{i=1}^{N}\prod_{t=1}^{n} K\left(\frac{x_t - x_{i,t}}{s_t}\right)\mathbf{y}_i}{\sum_{i=1}^{N}\prod_{t=1}^{n} K\left(\frac{x_t - x_{i,t}}{s_t}\right)}, \tag{3}$$

where $K(.)$ is a kernel with a bandwidth s.

Several types of kernel functions are commonly used, such as: uniform, triangle, Epanechnikov, biweight or Gaussian. The last one is used in experimental part of the work. The model parameters, bandwidths $s_1, s_2, \ldots, s_n$, are estimated in the grid search procedure on the training set (see [6] for training details).

B. Fuzzy Neighborhood Model

In the Fuzzy Neighbor Model (FNM) the fuzzy membership of the training pattern $\mathbf{x}_i$ to the neighborhood of the current input pattern $\mathbf{x}$ (query pattern) is defined as [6]:

$$\mu(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{d^2(\mathbf{x}, \mathbf{x}_i)}{s^2}\right), \tag{4}$$

where s is a bandwidth of Gaussian membership function (4) and $d(.)$ is a distance measure.

The membership function measures similarity between x-patterns. It is used as a weighting function in the regression model:

$$h^2(\mathbf{x}) = \frac{\sum_{i=1}^{N}\mu(\mathbf{x}, \mathbf{x}_i)\mathbf{y}_i}{\sum_{i=1}^{N}\mu(\mathbf{x}, \mathbf{x}_j)}. \tag{5}$$

The only parameter of FNM is bandwidth s. It is estimated easily in a grid search procedure [6].

C. K-Means Clustering K-M1

In the K-M1 model [6] the paired vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ are concatenated and form vector $\mathbf{u}_i = [\mathbf{x}_i^T\ \mathbf{y}_i^T]^T$. When we prepare a forecast for the day of type z (Monday, …, Sunday), the vectors $\mathbf{u}$ that include y-patterns representing day type z are selected and grouped using k-means algorithm. The cluster centroids $\mathbf{m}_j$ have two parts corresponding to x- and y-patterns: $\mathbf{m}_{x,j}$ and $\mathbf{m}_{y,j}$. In the forecasting phase the query x-pattern is presented and assigned to the cluster j^* represented by the closest centroid $\mathbf{m}_{x,j^*}$. The y-part of this centroid is the model output:

$$h^3(\mathbf{x}) = \mathbf{m}_{y,j^*}. \tag{6}$$

The number of clusters is adjusted during the learning phase.

D. K-Means Clustering K-M2

In this approach patterns $\mathbf{x}$ and $\mathbf{y}$ are grouped independently using k-means algorithm into K and L clusters, respectively [6]. For the forecasting task for day type z the pairs $(\mathbf{x}_i, \mathbf{y}_i)$ from the training set are selected which include y-patterns representing days of type z. Patterns from this subset are grouped and two populations of clusters are created, C_x and C_y, represented by centroids $\mathbf{m}_{x,j}$ and $\mathbf{m}_{y,j}$, respectively. Then pairs $(\mathbf{x}_i, \mathbf{y}_i)$ from the training subset are presented, and the empirical conditional probabilities $P(C_{y,l}|C_{x,k})$ that the forecast pattern $\mathbf{y}_i$ belongs to cluster $C_{y,l}$, when the corresponding input pattern $\mathbf{x}_i$ belongs to cluster $C_{x,k}$, are estimated. In the forecasting phase the query x-pattern is assigned to the group C_{x,i^*}. The forecasted y-pattern paired with it is determined from the prototypes $\mathbf{m}_y$ weighted by the conditional probabilities $P(C_{y,l}|C_{x,i^*})$:

$$h^4(\mathbf{x}) = \frac{\sum_{l=1}^{L}P(C_{y,l} \mid C_{x,i^*})\mathbf{m}_{y,l}}{\sum_{l=1}^{L}P(C_{y,l} \mid C_{x,i^*})}. \tag{7}$$

The number of clusters K and L are selected in the training procedure.

E. Artificial Immune System AIS1

In AIS1 [6] the concatenated patterns $\mathbf{x}$ and $\mathbf{y}$ are represented by antigens (AGs) with epitopes $\mathbf{u}_i = [\mathbf{x}_i^T\ \mathbf{y}_i^T]^T$. AGs are recognized by antibodies (ABs) which play a role of clusters. Epitopes correspond to AB paratopes (cluster centroids) which are built analogously to the epitopes: $\mathbf{v}_i = [\mathbf{p}_i^T\ \mathbf{q}_i^T]^T$, where $\mathbf{p} \in X = \mathbb{R}^n$ corresponds to x-patterns, and $\mathbf{q} \in Y = \mathbb{R}^n$ corresponds to y-patterns. The paratopes are modified during training.

Each AB has the recognition region in the form of n-dimensional hyperball of radius $r > 0$ with center in the point $\mathbf{p}_i$. The radius r, which is called the cross-reactivity threshold, is fixed, the same for each AB. The cluster represented by an AB contains those AGs, for which AB demonstrates nonzero affinity. The affinity depends on the distance between vectors $\mathbf{x}$ and $\mathbf{p}$ as well as on the AB cross-reactivity threshold.

In the training process the immune memory is created which includes a population of ABs covering a population of training AGs. During training an average forecast error is minimized. In the forecasting phase an incomplete AG is presented having only the x-part of the epitope. This AG is recognized by the memory ABs with positive affinity for it. The y-part of the AB epitope is reconstructed from the q-paratopes of activated ABs according to the formula:

$$h^5(\mathbf{x}) = \frac{\sum_{k\in\Theta} a(\mathbf{p}_k, \mathbf{x})\mathbf{q}_k}{\sum_{k\in\Theta} a(\mathbf{p}_k, \mathbf{x})}, \qquad (8)$$

where: $a(\mathbf{p}_k, \mathbf{x}) \in [0, 1]$ is the affinity of the k-th AB for the recognized AG and Θ is a set of indices of activated ABs.

F. Artificial Immune System AIS2

AIS2 includes two populations of ABs: population of type x (ABx), which recognize AGs representing x-patterns (AGx), and population of type y (ABy), which recognize AGs representing y-patterns (AGy) [6]. Patterns $\mathbf{x}$ are the epitopes of AGxs and paratopes of ABxs, and patterns $\mathbf{y}$ are the epitopes of AGys and paratopes of ABys. Both epitopes and paratopes are fixed. Each ABx has the cross-reactivity threshold r defining its recognition region: n-dimensional hyperball of radius r with center in the point $\mathbf{x}$. Similarly, ABy has the recognition region of radius s with center in the point $\mathbf{y}$. Radii r and s are adjusted individually during training, so that AB recognizes AGs having epitopes which are similar to the AB paratope. This similarity is expressed as an affinity (activation strength). One AG can activate many ABs. AB represents a cluster of similar AGs in the feature space X or Y. The k-th ABx can be seen as a pair $(\mathbf{p}_k, r_k)$, where $\mathbf{p}_k = \mathbf{x}_k$ is a paratope recognizing and representing AGx epitopes. The k-th ABy can be seen as a pair $(\mathbf{q}_k, s_k)$, where $\mathbf{q}_k = \mathbf{y}_k$ is a paratope recognizing and representing AGy epitopes.

After the two populations of the immune memory have been created, the empirical conditional probabilities $P(ABy_k \mid ABx_j)$, $j, k = 1, 2, \ldots, N$, that the i-th AGy activates the k-th ABy, when the corresponding i-th AGx activates the j-th ABx, are determined on the training population of AGs. These probabilities as well as the affinities are used to predict y-pattern according to the formula:

$$h^6(\mathbf{x}) = \frac{\sum_{j=1}^{N}\sum_{i\in\Theta} P(ABy_j \mid ABx_i) a(\mathbf{p}_i, \mathbf{x})\mathbf{q}_j}{\sum_{k=1}^{N}\sum_{i\in\Theta} P(ABy_k \mid ABx_i) a(\mathbf{p}_i, \mathbf{x})}. \qquad (9)$$

G. Artificial Immune System AIS3

In AIS3 each pair of patterns $(\mathbf{x}_i, \mathbf{y}_i)$ from the training set is represented by AG [7]. Some of the x-pattern components form the AG epitope corresponding to the AB paratope. One AG can have many epitopes as it can be bound by many ABs with different paratopes. An AG has a label containing a target y-pattern. An AB is composed of five elements: $\{\mathbf{p}, \mathbf{q}, \Omega, r, P\}$. A vector $\mathbf{p}$ correspond to a vector $\mathbf{x}$ and a vector $\mathbf{q}$ correspond to a vector $\mathbf{y}$. Vectors $\mathbf{p}$ and $\mathbf{q}$ of the k-th AB are initialized by k-th training example: $\mathbf{p}_k = \mathbf{x}_k$, $\mathbf{q}_k = \mathbf{y}_k$. A set Ω includes selected components of the vector $\mathbf{p}$ which form a paratope. The vector $\mathbf{q}$ is saved in the AB label. r is the cross-reactivity threshold. P is an AB power (representativeness) expressing how many training AGs it recognizes. Vectors $\mathbf{p}$ are fixed but vectors $\mathbf{q}$, paratopes Ω, thresholds r and powers P are modified during training.

Elements $\mathbf{p}$, Ω and r define the recognition region of AB, which is a hyperball of radius r centered at point $\mathbf{p}' = [p_t]_{t\in\Omega}$. Thus, the hyperball is defined in a subspace of the n-dimensional input feature space X. This subspace contains dimensions which are saved in the set Ω.

The AB recognition regions are adapted during training to the population of AGs. As a result of training the population of immune memory cells is created. A training AG is recognized by an AB when it is inside the AB recognition region and labels of AG and AB are similar. In the forecasting phase new AG with empty label is recognized by some ABs. The AG label is predicted from the labels of activated ABs, their powers and affinities expressing similarity between x-vector of AG and p-vectors of ABs:

$$h^7(\mathbf{x}) = \frac{\sum_{j=\Theta} P_j a(\mathbf{p}_j, \mathbf{x}, \Omega_j)\mathbf{q}_j}{\sum_{k\in\Theta} P_k a(\mathbf{p}_k, \mathbf{x}, \Omega_k)}, \qquad (10)$$

where: $a(\mathbf{p}_j, \mathbf{x}, \Omega_j) \in [0, 1]$ is the affinity of j-th AB for AG with epitope $\mathbf{x}$ defined in Ω_j-subspace.

H. ARIMA

To simplify the forecasting problem in ARIMA the load time series is decomposed into n series, i.e. for each period t a separate series was created. The model parameters are estimated using 12-week time series fragment immediately preceding the forecasted day. In this fragment only weekly seasonality is observed. The seasonal ARIMA model is built: ARIMA(p, d, q)×(P, D, Q)$_v$, where $v = 7$, i.e. one week period. The ARIMA parameters are estimated using the step-wise procedure for traversing the model space which is implemented in the **forecast** package for the **R** system for statistical computing [8]. In this automatic procedure the Akaike Information Criterion (AIC) is minimized.

I. Exponential Smoothing

For exponential smoothing (ExSm) the same time series decomposition is performed as for ARIMA. Model parameters are also estimated using 12-week time series fragment. In ExSm seasonal, trend and error components can be composed additively or multiplicatively, and the trend can be damped or not. The best ExSm model is selected using the automated procedure implemented in the **forecast** package for the **R** system [8]. In this procedure smoothing and damping parameters are estimated as well as the initial states of the level, growth and seasonal components. AIC is used for selecting the best model for a given time series.

J. Neural Model

The neural model is learned locally [9] using training patterns selected from the neighborhood of the query pattern **x**. By the neighborhood of **x** we mean the set of its k nearest neighbors representing the same day of the week (k is assumed to be 12). For each forecasting task a separate model is learned using Levenberg-Marquardt algorithm with Bayesian regularization to prevent overfitting. Local modeling of the target function using small numbers of training patterns implies small number of neurons. Based on the preliminary study [9], as an optimal architecture the neural model with only one sigmoid neuron is selected.

III. Ensemble Intergration

We are given a set of $M = 10$ individual learners $\{h^1, h^2, ..., h^M\}$. The forecasting tasks that each learner solves are: on the basis of input data predict $L_{i+\tau,t}$, i.e. the system load in period t of the day $i+\tau$. Let us denote the output of h^k for the specific forecasting task as $h^k_{i+\tau,t} \in$ R. We combine h^k's to attain the final prediction on the real-valued variable, i.e. the forecast of $L_{i+\tau,t}$. The easiest way of combining learners is simple averaging. The combined ensemble output $H_{i+\tau,t}$ is of the form:

$$H_{i+\tau,t} = \frac{1}{M}\sum_{k=1}^{M} h^k_{i+\tau,t} \,. \tag{11}$$

According to (11) each learner gets the same weight (1/M) which is independent on the forecasting task and the learner performance. More sophisticated approach assigns the weights to the learners taking into account the learner performance on the forecasting tasks similar to the current task. By similar forecasting tasks we mean these ones which satisfy the conditions:

- they have the same forecast horizon,
- type of the forecasted day z (Monday, ..., Sunday) is the same,
- forecasted days are from the same period of the year,
- forecasted period t is the same.

For example when the current task is to forecast load at hour 12, on Monday, with horizon 3, we deem as similar the forecasting tasks for hour 12 on m Mondays preceding the forecasted Monday (shifted in time 1, 2, ..., m weeks back) and with $\tau = 3$. The mean error of the k-th model on the similar forecasting tasks is used to determine its weight:

$$w^k_{i+\tau,t} = \exp\left(-\frac{(\overline{E}^k_{i+\tau,t})^2}{2\sigma^2_{i+\tau,t}}\right), \tag{12}$$

where $\overline{E}^k_{i+\tau,t}$ is the mean forecast error (MAPE) of the k-th model for the period t of the days $i-7+\tau$, $i-14+\tau$, ..., $i-7m+\tau$:

$$\overline{E}^k_{i+\tau,t} = \frac{1}{m}\sum_{j=1}^{m} E^k_{i-7\cdot j+\tau,t} \,, \tag{13}$$

$\sigma_{i+\tau,t}$ is the Gaussian curve (12) bandwidth defined as the median of the mean errors $\overline{E}^k_{i+\tau,t}$ for all ensemble members:

$$\sigma_{i+\tau,t} = median(\overline{E}^1_{i+\tau,t}, \overline{E}^2_{i+\tau,t}, ..., \overline{E}^m_{i+\tau,t}) \,. \tag{14}$$

The bandwidth $\sigma_{i+\tau,t}$ is dependent on the ensemble member errors achieved in the forecasting tasks similar to the current one. The models with mean errors lower than the median error have weights higher than exp(–0.5) = 0.6065. The maximal weight, equal to 1, will be assigned to the model with $\overline{E}^k_{i+\tau,t} = 0$. Note that weights $w^k_{i+\tau,t}$ are dynamically updated for each forecasting task. This is because for a new forecasting task we estimate the expected error on the similar forecasting tasks from the history. Then we use this error $\overline{E}^k_{i+\tau,t}$ as well as updated bandwidth $\sigma_{i+\tau,t}$ in weighting function (12). Thus, the models which were in the nearest past the most accurate in forecasting system load in the period t of a certain day of the week and horizon have the highest weights when we build a forecast for this day of the week, period and horizon.

The ensemble output when using dynamically updated weights is of the form:

$$H_{i+\tau,t} = \sum_{k=1}^{M} w'^k_{i+\tau,t}\, h^k_{i+\tau,t} \,, \tag{15}$$

where weights $w'^k_{i+\tau,t}$ are normalized versions of the weights $w^k_{i+\tau,t}$:

$$w'^{k}_{i+\tau,t} = \frac{w^{k}_{i+\tau,t}}{\sum_{l=1}^{M} w^{l}_{i+\tau,t}}, \quad (16)$$

such that they sum up to 1: $\sum_{l=1}^{M} w'^{l}_{i+\tau,t} = 1$.

IV. Simulation Study

The proposed heterogeneous ensembles were examined on four load time series:

- PL: time series of the hourly load of the Polish power system over the period 2002–2004. The test set includes data from 2004 with the exception of 13 atypical days (e.g. public holidays),
- FR: time series of the half-hourly load of the French power system over the period 2007–2009. The test set includes data from 2009 except for 21 atypical days,
- GB: time series of the half-hourly load of the British power system over the period 2007–2009. The test set includes data from 2009 except for 18 atypical days,
- VC: time series of the half-hourly load of the power system of Victoria, Australia, over the period 2006–2008. The test set includes data from 2008 except for 12 atypical days.

For all time series the training sets include data from the first two years. The problem is to forecast the system hourly (for PL) or half-hourly load (for FR, GB and VC) for the next seven days ($\tau = 1, 2, ..., 7$). The forecasting tasks are solved independently by each of ten forecasting models presented in Section II. Then model outputs are combined and ensembles are created: Ens1, using simple averaging (11) and Ens2, using dynamically weighted averaging for $m = 5$ (15).

In Fig. 1 the errors generated by the base models and ensembles for different horizons are presented. The most accurate models and their errors in Tables 1–4 are shown. The asterisks in this tables indicate the best models which are statistically indistinguishable for a given horizon (confirmed by Wilcoxon rank sum test with 5% significance level).

Note that combining forecasts of the base models in ensembles leads to reduction in forecast error. In comparison with individual base models the ensembles in both variants, Ens1 and Ens2, give better results in almost all cases: for each dataset and horizon. Also the variance of the forecasts is reduced. This is shown in Fig. 2 where interquartile ranges of errors IQR as a measure of error dispersion are presented. There is no significant difference in results between Ens1 and Ens2. The dynamic weighting in Ens2 did not bring an expected improvement in accuracy.

V. Conclusions

In this work heterogeneous ensembles composed of ten different forecasting models for short-term electricity demand forecasting are proposed. The forecasts generated by the base models are combined using simple averaging or dynamically weighted averaging. In the latter case the model weight is dependent on the model performance on the similar forecasting task to the current task.

As experimental study have shown the results for both ensemble approaches, with fixed and dynamic weights, brought improvement in results in comparison to individual models. The advantages of ensembles reported in literature, such as more certain, precise and accurate results, were confirmed. For each dataset and forecast horizon the ensemble errors were lower than the mean errors for the base models, and in most cases even lower than error of the best model. The variance of the forecasts when using ensembles was reduced as well. Both ensemble approaches gave similar results. Thus, adjusting the ensemble by dynamic weighting strategy did not

TABLE I. Best Results for PL Data

Horizon	Best model		2nd best model		3rd best model	
1	Ens2	1.25*	Ens1	1.27*	N-WE	1.30*
2	Ens2	1.72*	Ens1	1.74*	N-WE	1.90
3	Ens2	2.15*	Ens1	2.16*	N-WE	2.31
4	Ens2	2.48*	Ens1	2.49*	ExSm	2.67
5	Ens2	2.73*	Ens1	2.73*	ExSm	2.85
6	Ens2	2.89*	Ens1	2.89*	ExSm	2.98
7	Ens2	2.99*	Ens1	3.00*	ExSm	3.10

TABLE II. Best Results for FR Data

Horizon	Best model		2nd best model		3rd best model	
1	Ens2	1.59*	Ens1	1.60*	Neural	1.64
2	Ens1	2.54*	Ens2	2.54*	Neural	2.68
3	Ens2	3.20*	Ens1	3.21*	FNM	3.25*
4	Ens1	3.64*	Ens2	3.65*	N-WE	3.73
5	Ens1	4.12*	Ens2	4.14*	N-WE	4.17
6	Ens2	4.48*	Ens1	4.48*	N-WE	4.52
7	Ens2	4.65*	Ens1	4.65*	N-WE	4.67

TABLE III. Best Results for GB Data

Horizon	Best model		2nd best model		3rd best model	
1	Ens2	1.52*	Ens1	1.52*	N-WE	1.55*
2	Ens1	1.96*	Ens2	1.96*	N-WE	2.02
3	Ens1	2.28*	Ens2	2.28*	N-WE	2.33*
4	Ens1	2.58*	Ens2	2.58*	N-WE	2.62*
5	Ens1	2.77*	Ens2	2.77*	N-WE	2.81*
6	Ens1	2.99*	Ens2	3.00*	N-WE	3.03*
7	N-WE	3.22*	Ens1	3.24*	Ens2	3.25*

TABLE IV. Best Results for VC Data

Horizon	Best model		2nd best model		3rd best model	
1	Ens2	2.68*	Ens1	2.68*	N-WE	2.82
2	Ens2	3.76*	Ens1	3.79*	ExSm	4.02
3	ExSm	4.04*	Ens2	4.07*	Ens1	4.08*
4	ExSm	4.01*	Ens2	4.02*	Ens1	4.04
5	ExSm	3.99*	Ens2	4.00	Ens1	4.01
6	ExSm	4.00*	Ens2	4.12	Ens1	4.14
7	ExSm	4.15*	Ens2	4.23	Ens1	4.26

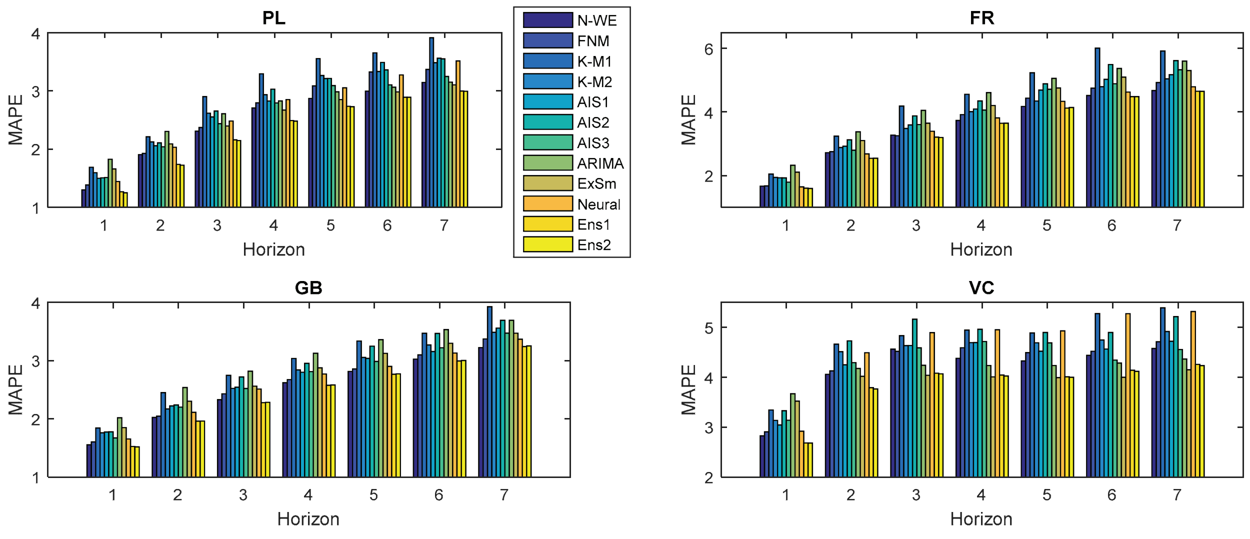

Fig. 1. Errors generated by base models and ensembles for different forecast horizons.

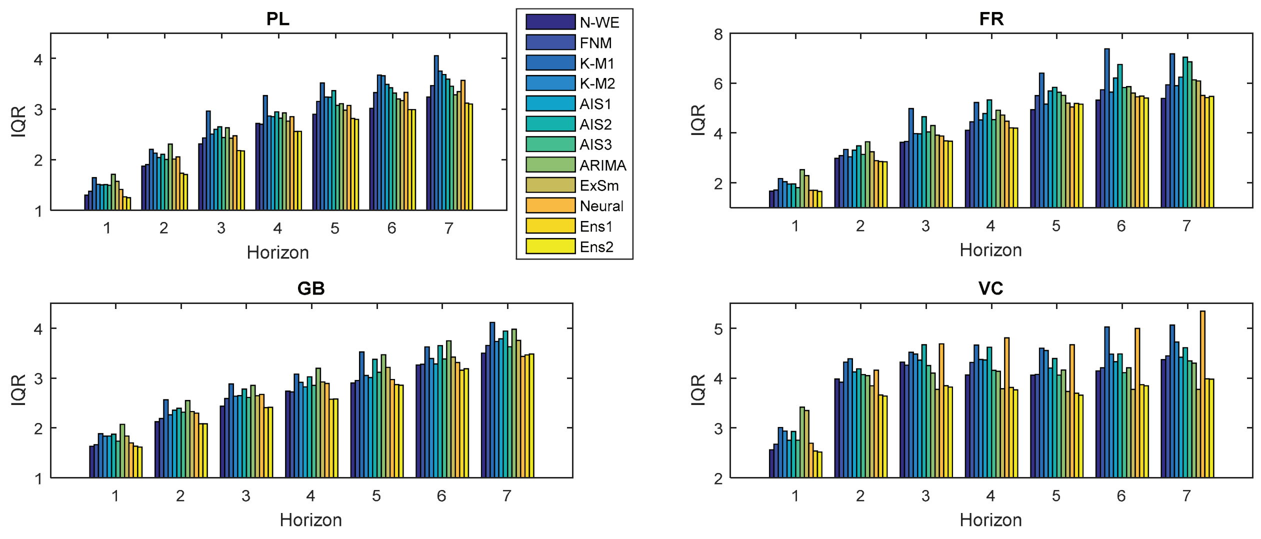

Fig. 2. Interquartile ranges of errors generated by base models and ensembles for different forecast horizons.

bring expected improvement in performance. However, the more general conclusions need more experimentation with weighting function parameters.

References

[1] J. Mendes-Moreira, C. Soares, A.M. Jorge, and J.F. D. Sousa, "Ensemble approaches for regression: a survey," ACM Computing Surveys, vol. 45, no.1, pp. 10:1–10:40, 2012.

[2] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, Advances in Neural Information Processing Systems 7, pp. 231–238. MIT Press, Cambridge, MA, 1995.

[3] Z.H. Zhu, Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC Data Mining and Knowledge Discovery Serie, 2012.

[4] J. Wichard, C. Merkwirth, and M. Ogorzałek, "Building ensembles with heterogeneous models," in Course of the International School on Neural Nets, 2003.

[5] G. Dudek, "Pattern similarity-based methods for short-term load forecasting – part 1: principles," Applied Soft Computing, vol. 37, pp. 277–287, 2015.

[6] G. Dudek, "Pattern similarity-based methods for short-term load forecasting – part 2: models," Applied Soft Computing, vol. 36, pp. 422–441, 2015.

[7] G. Dudek, Similarity-based Machine Learning Methods for Short-Term Load Forecasting. Academic Publishing House EXIT, Warsaw, 2012 (in Polish).

[8] R.J. Hyndman, Y. Khandakar, "Automatic time series forecasting: the forecast package for R," Journal of Statistical Software, vol. 27, no. 3, pp. 1–22, 2008.

[9] G. Dudek, "Forecasting time series with multiple seasonal cycles using neural networks with local learning," Proc. Artificial Intelligence and Soft Computing ICAISC 2013, LNCS 7894, Springer, pp. 52–63, 2013.