An ensemble stacked model with bias correction for improved water demand forecasting

Xenochristou, Maria; Kapelan, Zoran

DOI
10.1080/1573062X.2020.1758164

Publication date
2020

Document Version
Accepted author manuscript

Published in
Urban Water Journal

An ensemble stacked model with bias correction for improved water demand forecasting

Maria Xenochristou[a]* and Zoran Kapelan[b]

[a, b] Centre for Water Systems, University of Exeter, North Park Road, EX4 4QF Exeter, U.K.; [b] Delft University of Technology, Stevinweg 1, 2628CN Delft, Netherlands

*mx220@exeter.ac.uk

Abstract

Water demand forecasting is an essential task for water utilities, with increasing importance due to future societal and environmental changes. This paper suggests a new methodology for water demand forecasting, based on model stacking and bias correction that predicts daily demands for groups of ~120 properties. This methodology is compared to a number of models (Artificial Neural Network – ANN, Generalised Linear Model – GLM, Random Forest - RF, Gradient Boosting Machine - GBM, Extreme Gradient Boosting – XGBoost, and Deep Neural Network - DNN), using real consumption data from the UK, collected at 15-30 minute intervals from 1,793 properties. Results show that the newly proposed stacked model that comprises of RF, GBM, DNN, and GLM models consistently outperformed other water demand forecasting techniques (peak R^2 = 74.1%). The stacked model's accuracy on peak consumption days further improved by applying a bias correction method on the model's output.

Keywords: water demand forecasting; gradient boosting machines; machine learning; model stacking; bias correction; deep neural networks

Introduction

Satisfying the water supply-demand balance is a major challenge in many countries and a topic of increasing concern in the UK. According to the Government's water strategy for England report (Defra, 2008), which plans strategies for securing the future of water resources and improving the water environment till 2030, an essential aspect of managing

water demand is by ensuring a good forecasting of future patterns. However, forecasting demand is a challenging task, due to the nature and quality of the available data, the numerous factors that influence water consumption, as well as the various forecast horizons and spatial scales (Mamade *et al.*, 2014).

With the advancement in technology and computing power as well as the increasing data availability, machine learning has become a popular approach for water demand forecasting (Froukh, 2001; Cutore et al., 2008; Firat, Turan and Urdusev, 2009; Bai et al., 2014; Bakker et al., 2014; Romano and Kapelan, 2014; Shabani et al., 2016). There is currently an abundance of methods and models available, from the more researched Artificial Neural Networks (ANNs) to the relatively newer concept of ensemble machine learning.

ANNs have been used in many studies and have been proven effective to predict short-term, medium-term, and long-term water demand (Bougadis, Adamowski and Diduch, 2005; Adamowski, 2008; Firat, Turan and Urdusev, 2009; Herrera *et al.*, 2010; Dos Santos and Pereira, 2014; Mouatadid and Adamowski, 2017; Ghiassi, Fa'al and Abrishamchi, 2017; Altunkaynak and Nigussie, 2018). Adamowski (2008) used an ANN model to predict peak daily water demand for ~77,500 consumers in the city of Ottawa and found it performed better (R^2 = 69%) than multiple linear regression and time-series analysis. Dos Santos and Pereira (2014) tested eight model configurations of an ANN (3-layer, feed forward, back propagation) system for short-term water demand forecasting, using weather and temporal characteristics. The ANN was compared with multiple linear regression for hourly predictions for a large metropolitan area in Sao Paulo, Brazil. The best performance was obtained for the ANN that implemented 12-hour averages of the input variables and past consumption data as explanatory factors (R^2 = 67.9%). However,

the authors argued that the model could benefit from additional explanatory variables. Ghalehkhondabi et al. (2017) reviewed the water demand forecasting literature between 2005 and 2015 and concluded that although soft computing techniques have been extensively used, deep neural networks (DNN) have yet to be tested.

In recent years, some of the most successful models in machine learning competitions have been ensemble methods, which create a strong learner by combining multiple, individual, weak learners. There are three ensemble techniques, bagging, boosting, and stacking. Bagging is a resampling technique that randomly chooses a sub-sample of the dataset with replacement for training each learner (Mao, 1998). An example of a commonly used bagging algorithm is Random Forests (RFs) (Breiman, 2001), which are based on training multiple decision trees on different samples of the original training set. Boosting is also a resampling technique but in this case the instances of the training data that got misclassified from previous learners typically gain additional weight while the ones that were classified correctly lose weight. This way, the model gradually becomes better as it focuses on harder areas of the problem. Gradient Boosting Machines (GBMs) are an example of a commonly used machine learning algorithm that uses this method. Finally, stacking is the process of feeding the outputs of different machine learning models (base models) into one meta-learner (Ngo, Ernst, and Tokekar, 2018). Stacked models have been found to outperform individual models since they combine the strengths and reduce the weaknesses of their individual counterparts.

Although proven to perform better than their base models, ensemble models have been very rarely tested in water demand forecasting studies (Ghalehkhondabi *et al.*, 2017). Herrera *et al.* (2010) used RFs to forecast hourly urban water demand for a region of ~5,000 consumers and found them to perform worse than Support Vector Regression

(SVR), Multivariate Adaptive Regression Splines (MARS), and Projection Pursuit Regression (PPR). However, since not all parameters of the RFs were properly tuned, results could potentially improve. Tiwari, Adamowski and Adamowski (2016) assessed the capacity of extreme learning machines (ELMs) alone, or combined with wavelet analysis or bootstrap method and compared it with traditional ANN models. The aim was to forecast urban water demand for one day lead for the city of Calgary (~1.1 million consumers). The combined ELM wavelet (ELMw) model performed best for short-term forecasting and peak demands, with smaller errors and less computational time. However, there was a clear tendency in all models to over-predict the lower consumption days and under-predict the days with high consumption. Chen *et al.* (2017) also used RFs as well as a combined Wavelet Transform to predict daily water consumption for a supply area of 170,000 households and found that although the combined model performed better (R = 80%), it was still not capable of predicting the daily variations in water demand. Finally, Duerr *et al.* (2018) compared several time-series and machine learning models, including RFs and GBMs for monthly predictions at the household level and concluded that machine learning models generally underperformed when predicting monthly averages. However, the authors pointed out that improved data collection, high-resolution covariates, demographic information, as well as capturing the spatial dependence between neighbouring households could improve results.

As it becomes apparent from the above, although machine learning models have been commonly used for water demand forecasting, the classical, single methods cannot produce the most accurate results (Ghalehkhondabi *et al.*, 2017). Even when consumption is aggregated at a high temporal (e.g. monthly or quarterly) or spatial (e.g. city level) scale, the models commonly used in the literature struggle with accuracy, bias, and peak day predictions. Models based on deep learning, and ensemble

techniques, particularly model stacking, have been consistently found to produce excellent results in other fields. However, they have attracted very little to no attention in the water demand forecasting literature. Even when explored, essential aspects of the modelling and evaluation process like the tuning of the model's parameters or the assessment of the model's ability to perform on peak days and predict outliers are often overlooked.

This paper aims to address this gap by developing a new methodology based on model stacking and bias correction. This methodology is compared with a selection of ensemble and artificial neural network based models using real data from the UK. A detailed description of the data used in this study is provided in the next section. Then, the overall structure and characteristics of each model are described, followed by the bias correction methods. This includes details about the technical implementation of the models, such as the software, programming language, and open-source tools used. Next, the prediction accuracy of each model is presented under different forecasting scenarios (for all days as well as peak consumption days, with and without past consumption as input). The paper concludes with a discussion of the key findings, followed by a summary of results, conclusions, and recommendations for further research.

Data

An essential aspect of developing machine learning models is getting access to sufficient, high quality data. This study uses real data from the Southwest of England (Figure 1) that are available at very high temporal and spatial resolutions. Specifically, the dataset comprises of past consumption data and partial postcodes that became available by Wessex Water, one of the UK water companies, as well as weather data collected by the Meteorological Office of the United Kingdom (Met Office). Water

consumption data were collected at the household level using smart meters that recorded consumption every 15-30 minutes, over a period of 3 years (10/2014 – 9/2017), from 1,793 properties scattered around the study area. These data were cleaned and pre-processed in order to remove inconsistencies, errors, empty properties, and water-supply leakage. A more detailed description of this process is available in Xenochristou, Kapelan and Hutton (2020).

Water consumption was aggregated at the daily scale among houses with the same area postcode. The study area includes six postcodes, with up to 212 properties/day in each one, depending on data availability on the corresponding day and postcode (Figure 1). Since there are six postcodes and 1,019 days in the data, this resulted in 6,114 groups (6*1,019). Spatial analysis of the dataset concluded that smaller groups of properties are associated with increased forecasting errors (Xenochristou et al., 2020a), therefore groups with less than 60 properties were excluded from the data. As a result, the final dataset comprised of 5,063 groups with 120 properties/day on average.

The weather dataset included four weather variables, maximum air temperature, mean soil temperature at 10cm depth, mean relative humidity, and total rainfall. These data were recorded at the hourly or daily scale from hundreds of weather stations across the study area as part of the MIDAS (Met Office Integrated Data Archive System) dataset (Met Office, 2006a; Met Office, 2006b; Met Office, 2006c; Met Office, 2006d; Met Office, 2006e). In addition, the number of consecutive days without rain was calculated based on the daily rainfall. For each day in the data, one value for each weather variable was calculated as the weighted mean among the recorded values from multiple weather stations, based on their proximity to the properties in the study area. Weather stations that were located closer to the properties were assigned a higher weight whereas

weather stations with no households in close proximity (closer than any other weather station) were removed from the analysis. Weather records that had not been quality checked by the Met Office were also excluded.

Methodology

Model Inputs

All demand forecasting models developed here have a single output (or response) variable and a variety of inputs (or predictors). The predictor variables are a selection of potential explanatory factors that can influence water consumption and are thus used to explain part of the variance in the model. In this case, the response variable is the water consumption one day ahead, for a given postcode area. The predictor variables are past consumption data, area postcodes, as well as temporal and weather characteristics (Table 1).

Each input variable serves a different purpose into explaining demand variability. Water consumption is highly auto correlated from one day to the next one, therefore a sliding window of seven days (one input variable for each day) was chosen to capture the weekly repetition of water use. On the other hand, the location of a property is associated with a level of socio-economic status, property size, occupancy rate, garden size, property value, resident age, type of household (e.g. families with children, students, young professionals), or even certain habits. Since all of these customer and property characteristics are associated with different patterns and volumes of water use (Xenochristou et al., 2020b), the postcode was considered a valuable indicator of water habits. Finally, the temporal patterns of water use are a well-researched factor in the demand forecasting literature, therefore the type of day (working day vs

weekend/holiday) and the season were also used as model inputs (Bakker et al., 2013; Romano and Kapelan, 2014; Anele et al., 2017; Xenochristou et al., 2020b).

Previous work found that the weather's influence on water consumption varies over space and time (Xenochristou et al., 2018; Xenochristou, Kapelan and Hutton, 2020). Out of the six weather variables mentioned in the data section, the air and soil temperature as well as the rainfall and days without rain are highly correlated. Soil temperature has a lesser effect than air temperature and rainfall has little influence on water consumption in the UK (Xenochristou, Kapelan and Hutton, 2020), therefore these two variables were not used as model predictors. Only four weather variables, the number of sunshine hours, air temperature, humidity, and number of preceding, consecutive days without rain are used as model inputs (Table 1).

Based on the above, two model configurations are tested, one that includes all inputs (Group 1, Table 1) as well as one that excludes past consumption data (Group 2, Table 1). In terms of the practical value of this work, it is important to realise that many water utilities do not have access to high resolution consumption records. Even the ones that do, they do not have them in most cases for the whole extent of their network. Therefore, it is essential when evaluating the best model to also account for the model's ability to deal with the absence of past consumption data.

Model Tuning and Assessment

In machine learning, hyperparameters are the model parameters chosen before the training begins. The hyperparameter tuning step is a vital part of building an efficient machine learning model. It refers to defining a set of input parameters that influence the model structure and thus the results. The available parameters for tuning depend on the

type of model and can determine how closely the model fits on the training data. Fitting too closely could mean that the model learns from the noise in the training dataset (overfitting) which will result in a poor prediction on the testing dataset. On the other hand, fitting too loosely (underfitting) means that the model has not learnt to represent the patterns in the data.

Although there are different approaches to hyperparameter tuning (e.g. grid search, random search, evolutionary optimisation), in this study a random search as well as a simple grid search are used depending on the number of hyperparameters that need tuning at a time and the tools available. In a grid search, a number of values are defined for each parameter, creating a multi-dimensional grid space that includes every combination of hyperparameter values. Thus, when there is a high number of hyperparameters that need tuning, this approach can become time and computationally expensive. In a random search, the hyperparameter values are sampled from a pre-defined range of values. In both cases, each candidate model is built on a unique set of hyperparameters and the best model is chosen as the one that achieves the lowest mean square error (MSE) on the test dataset.

Initially, the dataset was shuffled and randomly divided into a training (70%) and a test (30%) dataset. The training set was used to fit the model and tune it for the optimum set of hyperparameters. The test set was used to perform an unbiased evaluation of the model's ability to perform predictions on unseen data, i.e. data that were not used during the model-building phase. In order to enhance the robustness of the hyperparameter selection, by ensuring their performance on different sets of data, a 5-fold cross validation process (Zhang, 1993) was implemented for the hyperparameter selection. This means that in every run, the

training data was shuffled and divided into five parts, out of which each time four were used for training and one for testing.

Three performance criteria were used to assess the model's performance: the mean absolute percentage error (MAPE), mean square error (MSE), and R^2 correlation coefficient, as each one of these provides slightly different, i.e. complementary information about the model's performance. The MAPE is one of the most common metrics, as it is easy to interpret and it scales the error in relation to the actual value. The MSE is sensitive to outliers, while the R^2 shows the variance in the dependent variable (model output) that can be explained by changes in the independent variables (model inputs) (Xenochristou, 2019).

Modelling Techniques

A number of modelling techniques such as neural networks, linear models, and representatives from every family of ensemble algorithms (bagging, boosting, and stacking) are compared in this study. All the model types that are used, either as a prediction tool or solely as a component of the stacked model are listed in the following.

Random Forests

Random Forests (RFs) were first introduced by Breiman (2001) as an ensemble of (hundreds or thousands) decision trees. The unique value of random forests is partly due to the implementation of randomness in the modelling process (Herrera *et al.*, 2010). A RF model trains each tree on a slightly different set of data, while at each split of the tree, it chooses among a different subset of input variables. The final result in regression is calculated as the mean prediction among all the trees in the forest. RFs have been consistently found to perform better than other machine learning techniques whilst

being a method that has not been fully explored in the water demand forecasting literature (Herrera *et al.*, 2010; Chen *et al.*, 2017).

There are three main parameters that need tuning in RFs, the mtry, ntrees, and tree depth (Scornet, 2017). The mtry is the number of variables randomly selected at each node and considered for splitting. Reducing the mtry increases the randomness of the tree-building process and therefore creates trees that are less similar to each other. The ntrees parameter is the number of trees used to build the forest. Model accuracy typically plateaus after a number of trees that are required to build a credible model. The tree depth is the point at which the tree stops growing, sometimes also denoted by the size of the final tree node (nodesize). The higher the tree depth, the closer the model fits on the training data, thus increasing the risk of overfitting. The RF model is tuned for the optimum values of all of the above three parameters.

Extremely Randomized Trees (XRT) are a variation of RFs that introduce added randomness in the above process. Similarly to RFs, a random subset of variables is selected for splitting at each node, but in this case a number of cutting-points (thresholds) are also selected at random. The best of these randomly selected thresholds is chosen for splitting at the node. The level of randomness implemented in the process can be tuned and is controlled by the model hyperparameters. In the extreme case, the trees are built completely at random, independent of the training sample (Geurts et al., 2006).

Gradient Boosting

Gradient Boosting Machines (GBMs) were first introduced by Friedman (2001) as an implementation of gradient boosting that explicitly deals with regression problems.

In the GBM implemented here, the base learner is also a decision tree. The boosting algorithm starts with one tree and at each iteration step, a new decision tree is fitted on the residuals of the previous tree and subsequently added to the model to update the residuals (Touzani et al., 2018). A shrinkage rate can also be applied on the algorithm, meaning that the new trees that are added to the model are gradually assigned lower weights. This increases the steps required for the algorithm to converge to a solution and reduces the risk of overfitting. The final result of the GBM is the weighted sum of the individual trees that were trained on weighted parts of the dataset (based on the accuracy achieved at the previous step).

A total of nine hyperparameters are tuned for the GBM model that aim to assist the algorithm arrive at the best solution, by implementing randomness in the modelling process or avoiding overfitting. In addition to the number of trees (ntrees), maximum tree depth (max_depth), and number of variables sampled for splitting (col_sample_rate), the number of variables sampled for each tree (col_sample_rate_per_tree) is also defined by the user. The number of variables sampled at each node is then calculated as the product of the variables sampled for the tree, multiplied by the variables sampled for splitting. The learning rate of the algorithm (learn_rate) is the factor by which the contribution of each consecutive tree is reduced compared to the previous tree. Another parameter (histogram_type) defines the type of histogram used to speed up the selection of the best splitting point at each node. The 'auto' histogram type means that the cutting points tested for splitting are chosen by dividing the range of values of each variable in equal steps. The subsample size (sample_rate) determines the size of the random sample used at each iteration. Smaller samples result in lower testing errors whereas larger samples generally improve the training accuracy. Finally, two hyperparameters determine if a further split in a tree will

occur, based on the minimum required relative improvement in squared error (min_split_improvement) and the minimum number of observations in a leaf node to allow further splitting (min_rows). More details regarding the implementation of the GBM algorithm can be found in Malohlava and Candel (2017).

Extreme Gradient Boosting (XGBoost) is another implementation of a boosting algorithm. It was introduced by Chen and Guestrin (2016), as 'an efficient and scalable implementation of the gradient boosting framework by Friedman (2001)' (Chen and He, 2015). XGboost aims to prevent overfitting as well as maximise the efficiency of computer resources (Fan *et al.*, 2018). According to Chen and Guestrin (2016), 17 out of the 29 winning solutions published by Kaggle, an online coding competition platform, used XGBoost either as a single model or as part of a stacked model.

The number of iterations (nround), the subsample size (subsample), maximum tree depth (max_depth), and fraction of explanatory variables sampled at each tree (colsample_bytree) are also hyperparameters of the XGBoost algorithm. In addition, the shrinkage rate (eta) determines the learning rate of the algorithm in the training step, i.e. the amount by which the contribution of each consecutive tree is reduced compared to the previous tree. Additional parameters that need tuning for this algorithm are the gamma and min_child_weight that determine how conservative the algorithm is in terms of further partitioning at a leaf node. The larger these parameters, the more conservative the algorithm. More details about the implementation of the XGBoost package can be found in Chen and Guestrin (2016).

Artificial Neural Networks

Artificial Neural Networks (ANNs) are a family of machine learning algorithms inspired by nature, specifically biological neural networks, and are comprised of nodes, organised in layers. Each node receives information with a certain weight from another node or external stimuli, transforms it, and passes it to the next node or transfers it as external output. Nodes that belong in the same layer work collectively within the same depth of the network. The higher the number of layers, the deeper the ANN. The ANN implemented here is a feed-forward, single hidden layer neural network. It was tuned for the number of units in the hidden layer (size) as well as a gradient decay (decay), i.e. a factor less than 1 by which the weights are multiplied at each update, i.e. each iteration of the algorithm.

Deep Neural Networks (DNNs) are ANNs composed of multiple layers, which allows them to transform information and learn from data with multiple abstraction levels (LeCun et al., 2015). The DNN implemented here is a multi-layer, feedforward ANN trained using stochastic gradient descent and back-propagation (Candel et al., 2014). In back-propagation, the model's error is fed back into the model in order to update the weights and further improve results. This process evolves as an optimisation problem, where the objective is to minimise the model's error using stochastic gradient descent (Bottou, 2010).

Although there are many parameters in a DNN, the following eight are tuned in this study. The number of epochs indicates how many times the whole dataset, divided into smaller batches, will go back and forth through the neural network during the training process. The higher the number of epochs, the higher the risk of overfitting while too few could lead to underfitting. The activation function (activation) transforms the input in a node to a certain output, while the size of the hidden layers (hidden) determines the

number of nodes in each one. The dropout ratio of the input (input_dropout_ratio), as well as the dropout ratio of the hidden layers (hidden_dropout_ratio) aim to prevent model overfitting. At each training example, they suppress the activation of the nodes (in the input or hidden layers, respectively) by a certain probability (dropout ratio). As a result, each training example creates a different model. The combination of these learners resembles an ensemble model (Candel et al., 2014). There is also the option to activate an adaptive learning rate (adaptive_rate) method for gradient descent that determines how quickly the algorithm converges to an optimum solution. In this case, the momentum of the learning rate is determined by two more hyperparameters, the rho and epsilon (Candel et al., 2014). More information regarding the tuning parameters of a DNN can be found in Candel et al. (2014).

Generalised Linear Models

Generalised Linear Models (GLMs) are an extension of simple linear models for errors that do not follow the normal distribution or predictors whose influence is not linear (Aiello *et al.*, 2016). GLMs typically create regression models that follow an exponential distribution (Aiello *et al.*, 2016). There are two parameters tuned for the GLM, one that defines how the model deals with missing values and the alpha regularization parameter. The value of alpha determines the penalization function used to avoid model overfitting, reduce the variance in the error, and deal with correlated predictors (h2o.ai, 2019a). More information regarding the meaning of these two parameters can be found in Nykodym *et al.* (2019)

Model Stacking

Stacking is the process of combining the results of individual learners into one super-learner. The way of combining them could be a simple weighted average or using a machine learning model such as a RF or ANN to learn the best combination based on the residual errors.

Bias correction methods

The concept of model bias is well-documented in the machine learning literature (Zhang and Lu, 2012; Nguyen, Huang and Nguyen, 2015; Song, 2015; Ghosal and Hooker, 2018; Hooker and Mentch, 2018). Especially in methods such as RFs, where the final prediction is estimated as the mean among the predictions of the individual trees, the range of the prediction becomes smaller due to averaging compared to the actual range in the response variable. This leads to overestimating the smaller values and underestimating the larger values in the dataset. As opposed to the above, which is a fundamental statistical concept, the systematic bias in the model's results refers to a consistent overprediction or underprediction of the response variable. A well-performing model should ideally exhibit a zero or near-zero systematic bias.

In this paper, four methods for bias correction (BC) described in Song (2015) are tested for their ability to improve the accuracy of the final result. In the first BC method (BC1), a RF model is used to predict the residual errors based on a set of predictors in the training dataset that include the predicted values of the response variable. The final prediction of the model is then adjusted by adding the predicted residuals to the predicted outcome. In the second BC method (BC2), a simple linear model is fitted on the residuals of the training set but this time only the predicted values are used as input. The same linear model is then used to predict the residuals in the test dataset. As with BC1, the residuals are added in the final prediction to adjust it. BC methods 3 and 4

(BC3 and BC4) use a residual rotation approach. They first calculate the prediction and the residuals based on Method 1. Then a simple linear model is fitted on the residuals against the predicted values. In BC method 3, the residuals are rotated so that y=0, whilst in BC method 4 the best rotation angle is determined sequentially as the one that achieves the minimum MSE. An extensive description of the four methods can be found in Song (2015). The code used for the implementation of the four BC methods is adapted by Song (2015).

Model Technical Implementation

All models, analysis, and results produced in this work were created using R (R core team, 2013). The RF, XGBoost, and ANN models were trained using the algorithms implemented in the 'randomForest' (Liaw and Wiener, 2018), 'xgboost' (Chen et al., 2019), and 'nnet' (Ripley and Venables, 2016) packages, respectively, and tuned using 'caret' (Kuhn, 2019), which allows to perform a grid search for the optimum hyperparameter values. The GBM, DNN, GLM, and stacked models were built using an open source machine learning platform, 'h2o', and specifically its automated machine learning capability ('autoML'), which was accessed through an R interface using package 'h2o' (LeDell et al., 2019). This method was implemented for its high performance, speed, automation, and efficiency.

The automatic capability 'autoML' of the 'h2o' platform currently provides support for the implementation of five machine learning methods, RF, XRT, GBM, DNN, GLM, and in some cases also for the XGBoost algorithm, which was not available here. However, it only tunes the GBM, DNN, and GLM models using random search, whereas it uses default versions of the XRT and RF models (h2o.ai, 2019b). In addition to this, 'h2o autoML' also trains two stacked ensemble models. The first one is based on

the best combination among all trained models, including multiple models from the same type (e.g. RF) that were trained as part of the hyperparameter tuning process. The second stacking technique includes only the best model of each type (h2o.ai, 2019b). The metalearner algorithm used to combine the models in the automated machine learning capability of 'h2o' is a GLM model with non-negative weights. The weights are determined based on the combination that produces the best stacked model, i.e. the one that achieves the lowest error (h2o.ai, 2019b). Only the three properly tuned 'h2o' models (GBM, DNN, and GLM) are presented in the results section, although both the default XRT and RF were used as components to build the final stacked models.

Results

In this section, the forecasting performance of seven models (RF, XGB, GBM, GLM, ANN, DNN, and stacked) is compared based on four evaluation metrics, the MAPE for all days as well as peak days, the R^2, and the MSE. For comparison, the error of a simple model that assumes forecasted consumption for each day is equal to the mean consumption among all days in the dataset, was 10.1% for all days and 19.8% for the peak days, i.e. the 10% of the days with the highest consumption.

Only the two best models of each type (one with and one without past consumption as input) after the tuning process was completed are presented in the following. In addition to this, four bias correction (BC1-BC4) methods are applied on top of the best performing model. Group 1 (Table 2) summarises the results of the models that include past consumption whereas Group 2 (Table 2) demonstrates the results achieved by the models that include only postcode location, temporal, and weather characteristics as input. The final hyperparameter values for each model are summarised in the Appendix.

According to Group 1 (Table 2), when past consumption is included as input, the model with the best performance (R^2 = 74.1%, MAPE = 4%) is the stacked model created by 'h2o' as an ensemble of five individual learners (the best from each family). Specifically, the stacked model comprises of a GBM, XRT, GLM, DRF, and DNN model, with a corresponding contribution to the output of 31%, 24%, 19%, 14%, and 12%, respectively. Out of the four BC methods tested here, the second method (BC2, Table 2), which predicts residual errors based on the predicted value of the response variable, performs best. Although applying the BC2 method on top of the stacked model's results did not improve the overall model performance, it did reduce the MAPE of the peak days from 5.1% to 4.6% (Model 7 & 9, Group 1, Table 2). Out of the rest, the GBM (R^2 = 74.1%, MAPE = 4.1%) and RF (R^2 = 72.8%, MAPE = 4.1%) models have the highest forecasting accuracy for all days in the data. The neural network based models have the lowest peak day errors, with a MAPE of 4.8% for the ANN and 5.2% for the DNN. However, the ANN model does not perform as well for the other two performance metrics (R^2 = 70.8%, MSE = 55). This implies that the reason that the model performed better for peak days might be that it systematically overpredicts consumption, especially due to the high MSE value, which is an indicator of bias in the model. Finally, the worst performing model across most metrics (R^2 = 70.6%, MAPE = 4.2%, MSE = 55) is the GLM.

When past consumption is not included as input, the best performing model is again the stacked model (MAPE = 4.3% for all days and 5.1% for peak days, R^2 = 71.2%, MSE = 54). This time, it comprises of a GBM, DNN, DRF, GLM, and XRT model, with a percentage contribution to the output of 53%, 15%, 11%, 11%, and 10%, respectively. Adding BC2 further reduces the MAPE to 5.1% for peak days. The second best performing model in this case is again the GBM, which has the same MAPE for all days

(R^2 =70.9%, MAPE = 4.3%) and slightly higher (MAPE = 5.6%) for peak days. It is worth noting that the ANN model, which performes relatively well with past consumption input and is the model most commonly used in the literature, underperforms in this case (MAPE = 4.7% for all days and 5.8% for peak days, R^2 = 65.1%, MSE = 65). Similar results apply for the GLM model, which performs reasonably well with past consumption data (MAPE = 4.2% for all days and 5.8% for peak days, R^2 = 70.6%, MSE = 55), but whose error increases significantly without (MAPE = 4.7% for all days and 6.8% for peak days, R^2 = 63.8%, MSE = 67).

Figure 2 demonstrates an example of the actual versus predicted values for two model types, the GLM and Stacked-BC2 (Stacked with Bias Correction method 2) model, without past consumption data. According to Figure 2, the days with the lowest consumption are most of the times ovrepredicted, while the days with unusually high consumption are underpredicted. Although this effect is particularly prominent for the GLM (plot a, Figure 2), it improves significantly in the case of the Stacked-BC2 model (plot b, Figure 2).Overall, predicting demand becomes slightly more challenging when past consumption data is not available as well as for peak days (Peak days, Table 2). However, certain models are able to deal significantly better with the lack of additional information (e.g. XGB, GBM) compared to others (ANN, GLM). The method that seems to be affected the most with forecasting demands without past consumption is the method that is frequently suggested as best in the literature - the ANN model. The MAPE of this method increases for the peak days from 4.8% to 6%, i.e. by 25% when comparing the cases with and without past consumption data as input.

Finally, although slight differences exist, most models achieve very similar results for all days in the data, with a range in MAPE between 0.2% (with past consumption) and

0.4% (without past consumption) across the test dataset (MAPE – All days, Table 2, Groups 1 & 2, respectively). However, the range of errors increases significantly for peak days, i.e. the 10% of the days with the highest consumption, with a range in MAPE between 1.4% (with past consumption) and 1.7% (without past consumption) (MAPE – Peak days, Table 2, Groups 1 & 2, respectively).

Discussion

One of the main observations of this study is the power of stacked models to improve the prediction accuracy of their counterparts by adding up their individual strengths and overcoming their weaknesses. However, there is a time and cost sacrifice in exchange for improving accuracy. No machine learning technique is universally best for all types of data, purposes, and datasets. Therefore, it is important to account for the computational power, effort, as well as expertise that is required to identify and tailor the machine learning technique that will produce the best outcome.

Another important point is the level of transparency and interpretability associated with each model. Generally, the fewer the number of model parameters and the simpler the model, the easier it is to understand, explain, and interpret. According to Molnar (2019), transparency refers to understanding how the algorithm learns from the data and is independent of the trained model, whereas interpretability is the knowledge of how the model makes decisions, based on its features, weights, and parameters. A linear regression model is transparent as the way the algorithm is built is thoroughly explored and understood and at the same time it is interpretable, as the weight of each predictor indicates its influence on the response variable. DNNs on the other hand are neither transparent, nor directly interpretable due to the complexity and number of hyperparameters and hidden layers. RFs on the other hand are easier to interpret and

explain as they are essentially an ensemble of decision trees. For example, the closer a variable is to the root of the tree, the higher its importance. Stacked models can achieve high accuracy as they combine the strengths of different models but at the same time they have limited interpretability, as they lack a model structure. In some cases, sacrificing some accuracy in order to increase the level of understanding is the preferred solution. However, model-agnostic interpretability methods (i.e. ones that can be used on any machine learning model) can assist with overcoming this issue by combining the high accuracy of stacked models with the understanding of more transparent and interpretable methods (Molnar, 2019; Xenochristou et al., 2020b).

An interesting concept that has not been highlighted in previous water demand forecasting attempts is the concept of bias towards the mean. This is a combination of the elementary statistical concept of regression towards the mean and certain model structures (e.g. RF), prone to create biased results (Zhang and Lu, 2012). Regression towards the mean is the term for a statistical phenomenon that can be illustrated by a simple example as follows. For an extreme measurement of a variable, e.g. an unusually high daily temperature, it is unlikely that a second measurement would result in a similar or higher value. The most likely scenario is that the second measurement is going to be closer to the mean annual temperature. Another example described by Stigler (1997) is a student that scored really high at a test. In order for this high score to occur, it is likely that not only skill, but also luck was involved, a factor that might diminish if another test was taking place, resulting in a lower score. A similar concept can be applied to water demand. In order for a very high consumption to occur on a certain day for a population of 120 households, a number of factors need to contribute. For example, previous research (Xenochristou, Kapelan and Hutton, 2020) indicated that an affluent area on a Saturday with high air temperature, is likely to result in a high

consumption. However, there are a number of additional factors that will determine how high consumption is going to be on that day. This means that although days with the same weather characteristics, the same past consumption, in the same location, are likely to have a higher than normal consumption, for only one of these days consumption will be high enough to be an outlier in the data. As the model learns from all days that have the same characteristics, but not as an extreme consumption, the predictions are likely to gravitate towards mean values, which will naturally result in overpredicting and underpredicting the highest and the lowest values in the dataset, respectively. This effect is exaggerated in certain models such as RFs due to their structure, which is based around averaging among hundreds or thousands of individual predictions. Stacked models, on the other hand, are able to deal with outliers much better. In addition, a simple bias correction technique could achieve an additional reduction in errors for the days with the highest consumption. Therefore, choosing wisely the model structure and performing a bias correction on the results could significantly improve predictions on critical days.

Finally, this research demonstrates how a simple tool, 'h2o.ai', can assist with the water demand forecasting model development process. As machine learning becomes the mainstream approach in many sectors, there is an increasing need for people that are not trained in the field of computer science to use these tools efficiently. The 'h2o' platform can be useful not only in order to choose the best algorithm but also in order to efficiently tune the model's hyperparameters. One of the problems that were identified in previous studies is the lack of proper tuning of the machine learning algorithms that are used for forecasting. In addition, creating a grid space for hyperparameter tuning is a brute-force approach that is time-consuming and not computationally efficient for high-dimensional problems, even when it is parallelised. Using the 'autoML' function of

'h2o' for choosing the model hyperparameters, even when the preferred algorithm is known, could significantly reduce complexity, computational time, as well as improve the model's results.

Summary and Conclusions

This study explored the potential of a stacked ensemble model with added bias correction (BC) to produce improved water demand forecasts. In order to achieve maximum accuracy, a fine-tuning process was adopted. The potential of automating this process using the machine learning platform 'h2o' were explored and compared to model development using methods that require extensive user engagement and expertise. The proposed model was compared with several traditional (e.g. ANN) as well as more recent machine learning methods (e.g. DNN, GBM, XGB) in the water demand forecasting literature and was found to consistently outperform them, especially for peak days when past consumption data was not available.

Results show that the stacked-BC2 model performed best as it consistently achieved the best demand forecasting accuracy for both consumption cases (all days and peak days) and for both prediction model configurations (with and without past consumption as input). The MAPE of the stacked-BC2 model was 4% for all days and 4.6% for peak days when past consumption data was included as input, as opposed to 4.3% and 5.1%, respectively, when past consumption data was not available.

The GBM model had a similar prediction accuracy (MAPE = 4.1% for all days and 5.4% for peak days), especially when past consumption data was not available (MAPE = 4.3% for all days and 5.6% for peak days). At the same time, the GBM model turned out to be quicker and easier to build since it requires tuning only one set of parameters

whereas the stacked-BC2 model requires the development and tuning of multiple individual learners that are combined to create a super-learner. On top of all this, the GBM model has a higher level of transparency and interpretability, since the stacked-BC2 is built as an ensemble of different base models and thus it lacks a model structure. This means that in situations where demand forecasting accuracy is not of the utmost importance, the GBM model is a viable alternative to the stacked model.

Depending on the scenario, in terms of the data availability and forecasting goal, the choice of model could significantly alter results. For easier tasks (e.g. when past consumption data is available and when the focus is not on predicting outliers) most models perform well. However, in situations where data availability is limited and the goal is to predict days with abnormal consumption, different models produce a wider range of accuracy. Specifically, when predicting demand using past consumption data over all days in the dataset, all models perform similarly, with a range in MAPE from 4.0% to 4.2%. However, when focusing on harder aspects of the same problem, e.g. when past consumption data is not available and for peak consumption days, the MAPE among different models varies from 5.1% (stacked-BC2) to 6.8% (GLM), an increase of 33%.

Finally, this study concluded that applying simple techniques like bias correction on top of the model's results could improve predictions for the peak days. Although most demand forecasting models are accurate (achieving regularly MAPE lower than 5%), they struggle to predict outliers. This fact could be particularly problematic in the context of water demand forecasting since days with unusually high consumption are usually the critical ones for water utilities, a problem that could intensify with future changes in the climate. This technique, although it did not alter the overall accuracy of

the model, it improved predictions for the 10% of the days with the highest consumption (Table 2).

Although the above models were tested under two scenarios, with and without past consumption data, as well as for peak consumption days, it is not clear how the models would perform with a less rich or more noisy dataset. An uncertainty analysis around the amount and quality of data necessary for each model type to perform well is needed to assess the model's robustness and suitability to produce accurate forecasts under different data availability scenarios.

Overall, this paper focused on identifying models and techniques that can be used to improve predictions in water demand forecasting. However, this analysis was performed at a certain spatial and temporal scale. Future work will explore what is the best accuracy that can be achieved at different spatial scales as well as specifically assess the contribution of several input factors (weather, temporal, and household characteristics) on the model's prediction accuracy at each scale.

Acknowledgments

This study was funded as part of the Water Informatics Science and Engineering Centre for Doctoral Training (WISE CDT) under a grant from the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/L016214/1. The authors would like to thank Wessex Water and Chris Hutton for providing the data for this study. The code used for the bias correction was adjusted from the paper of Song (2015).

References

Adamowski, J. F. 2008. 'Peak Daily Water Demand Forecast Modeling Using Artificial

Neural Networks.' *Journal of Water Resources Planning and Management* 134(2): 119–128. doi: 10.1061/(asce)0733-9496(2008)134:2(119).

Aiello, S., Eckstrand, E., Fu, A., Landry, M., and Aboyoun, P. 2016. 'Machine Learning with R and H2O: Seventh Edition.'

Altunkaynak, A., and Nigussie, T. A. 2018. 'Monthly water demand prediction using wavelet transform, first-order differencing and linear detrending techniques based on multilayer perceptron models.' *Urban Water Journal* 15(2): 177–181. doi: 10.1080/1573062X.2018.1424219.

Anele, A. O., Hamam, Y., Abu-Mahfouz, A. M., and Todini, E. 2017. 'Overview, comparative assessment and recommendations of forecasting models for short-term water demand prediction.' *Water* 9(11). https://doi.org/10.3390/w9110887.

Bai, Y., Wang, P., Chuan, L., Xie, J., and Wang, Y. 2014. 'A multi-scale relevance vector regression approach for daily urban water demand forecasting.' *Journal of Hydrology* 517: 236–245. doi: 10.1016/j.jhydrol.2014.05.033.

Bakker, M., van Duist, H., van Schagen, K., Vreeburg, J., and Rietveld, L. 2014. 'Improving the performance of water demand forecasting models by using weather input.' *Procedia Engineering* 70: 93–102. doi: 10.1016/j.proeng.2014.02.012.

Bakker, M., Vreeburg, J., van Schagen, K., and Rietveld, L. 2013. 'A fully adaptive forecasting model for short-term drinking water demand.' *Environmental Modelling & Software* 48: 141-151. doi: 10.1016/j.envsoft.2013.06.012.

Bottou, L. 2010. 'Large-scale machine learning with stochastic gradient descent.' International Conference on Computational Statistics, 177–187.

Bougadis, J., Adamowski, K., and Diduch, R. 2005. 'Short-term municipal water demand forecasting.' *Hydrological Processes* 19(1): 137–148. doi: 10.1002/hyp.5763.

Breiman, L. 2001. 'Random forests.' *Ensemble Machine Learning: Methods and Applications* 157–175. doi: 10.1007/9781441993267_5.

Candel, A., Parmar, V., LeDell, E., and Arora, A. 2014. 'Deep Learning with H2O.' 1–21.

Available from : https://www.h2o.ai/wp-content/themes/h2o2016/images/resources/DeepLearningBooklet.pdf

Chen, G., Long, T., Xiong, J., and Bai, Y. 2017. 'Multiple Random Forests Modelling for Urban Water Consumption Forecasting.' *Water Resources Management* 31(15): 4715–4729. doi: 10.1007/s11269-017-1774-7.

Chen, T. and Guestrin, C. 2016. 'XGBoost: A scalable tree boosting system.' Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 785–794. doi: 10.1145/2939672.2939785.

Chen, T. and He, T. 2015. 'xgboost : eXtreme Gradient Boosting.' R package version 0.4-2, pp. 1–4.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., et al. 2019. Package 'xgboost'. R version 0.90.0.2.

Cutore, P., Campisano, A., Kapelan, Z., Modica, C., and Savic, D. 2008. 'Probabilistic prediction of urban water consumption using the SCEM-UA algorithm.' *Urban Water Journal* 5(2): 125–132. doi: 10.1080/15730620701754434.

Defra (2008). Future Water The Government's water strategy for England. Department for Environment, Food and Rural Affairs. Retrieved from: www.tsoshop.co.uk [Accessed 27 September 2019].

Duerr, I., Merrill, H. R., Wang, C., Bai, R., Boyer, M., Dukes, M. D., and Bliznyuk, N. 2018. 'Forecasting urban household water demand with statistical and machine learning methods using large space-time data: A Comparative study.' *Environmental Modelling and Software*. 102: 29–38. doi: 10.1016/j.envsoft.2018.01.002.

Dos Santos, C. C. and Pereira Filho, A. J. 2014. 'Water Demand Forecasting Model for the Metropolitan Area of São Paulo, Brazil', *Water Resources Management* 28(13): 4401–4414. doi: 10.1007/s11269-014-0743-7.

Fan, J., Wang, X., Wu, L., Zhou, H., Zhang, F., Yu, X., Lu, X., and Xiang, Y. 2018. 'Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting

daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China.' *Energy Conversion and Management* 164: 102–111. doi: 10.1016/j.enconman.2018.02.087.

Firat, M., Turan, M. E., and Yurdusev, M. A. 2009. 'Comparative analysis of fuzzy inference systems for water consumption time series prediction.' *Journal of Hydrology* 374(3–4): 235–241. doi: 10.1016/j.jhydrol.2009.06.013.

Friedman, J. (2001) 'Greedy Function Approximation : A Gradient Boosting Machine.' *The Annals of Statistics* 29(5): 1189-1232. doi: 10.1214/009053606000000795.

Froukh, M. L. 2001. 'Decision-support system for domestic water demand forecasting and management.' *Water Resources Management* 15(6): 363–382. doi: 10.1023/A:1015527117823.

Geurts, P., Ernst, D., and Wehenkel, L. 2006. 'Extremely randomized trees'. *Machine Learning* 63(1): 3–42. https://doi.org/10.1007/s10994-006-6226-1.

Ghalehkhondabi, I., Ardjmand, E., Young II, W. A., and Weckman, G. R. 2017. 'Water demand forecasting: review of soft computing methods.' *Environmental Monitoring and Assessment* 189(7). doi: 10.1007/s10661-017-6030-3.

Ghiassi, M., Fa'al, F. and Abrishamchi, A. 2017. 'Large metropolitan water demand forecasting using DAN2, FTDNN, and KNN models: A case study of the city of Tehran, Iran', *Urban Water Journal* 14(6): 655–659. doi: 10.1080/1573062X.2016.1223858.

Ghosal, I., and Hooker, G. 2018. 'Boosting Random Forests to Reduce Bias; One-Step Boosted Forest and its Variance Estimate'. Available at: http://arxiv.org/abs/1803.08000.

H2O.ai. 2019a. 'Appendix A -Parameters: alpha.' Available from: https://h2o-release.s3.amazonaws.com/h2o/master/3908/docs-website/h2o-docs/data-science/algo-params/alpha.html#description [Accessed 6 September 2019].

H2O.ai. 2019b. 'AutoML: Automatic Machine Learning'. Available from: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html [Accessed 6 September

2019].

H2O.ai. 2019c. 'Overview.' Available from: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html [Accessed 6 September 2019].

Herrera, M., Garcia-Diaz, J. C., Izquierdo, J., and Perez-Garcia, R. 2010. 'Predictive models for forecasting hourly urban water demand', *Journal of Hydrology* 387(1–2): 141–150. doi: 10.1016/j.jhydrol.2010.04.005.

Hooker, G., and Mentch, L. 2018. 'Bootstrap bias corrections for ensemble methods.' *Statistics and Computing* 28(1): 77–86. doi: 10.1007/s11222-016-9717-3.

Kuhn, M., 2019. Package 'caret'. R version 6.0-84.

Nykodym, T., Kraljevic, T., Wang, A., and Wong, W. 2015. 'Generalized Linear Modeling with H2O'.

LeCun, Y., Bengio, Y., and Hinton, G. 2015. 'Deep learning', Nature, 521(7553): 436–444. doi: 10.1038/nature14539.

LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T., et al. 2019. Package 'h2o'. R package version 3.26.0.2.

Liaw, A., and Wiener, M. 2018. Package 'randomForest'. R version 4.6-14.

Malohlava, M., and Candel, A. 2017. 'Gradient Boosting Machine with H2O.' Seventh Edition. Available at: http://h2o.ai/resources/.

Mamade, A., Loureiro, D., Covas, D., Coelho, S. T., and Amado, C. 2014. 'Spatial and temporal forecasting of water consumption at the DMA level using extensive measurements.' *Procedia Engineering* 70: 1063–1073. doi: 10.1016/j.proeng.2014.02.118.

Mao, J. 1998. 'Case study on Bagging, Boosting, and Basic ensembles of neural networks for OCR.' IEEE International Conference on Neural Networks - Conference Proceedings, 3: 1828–1833.

Met Office, N. B. A. D. C., 2006a. MIDAS: U.K. Daily Rainfall Data, s.l.: s.n.

Met Office, N. B. A. D. C., 2006b. MIDAS: U.K. Daily Temperature Data, s.l.: s.n.

Met Office, N. B. A. D. C., 2006c. MIDAS: U.K. Daily Weather Observation Data, s.l.: s.n.

Met Office, N. B. A. D. C., 2006d. MIDAS: U.K. Hourly Weather Observation Data, s.l.: s.n.

Met Office, N. B. A. C., 2006e. MIDAS: U.K. Soil Temperature Data, s.l.: s.n.

Molnar, C. 2019. 'Interpretable machine learning, A guide for making black box models explainable.' https://christophm.github.io/interpretable-ml-book/.

Mouatadid, S., and Adamowski, J. 2017. 'Using extreme learning machines for short-term urban water demand forecasting.' *Urban Water Journal* 14(6): 630–638. doi: 10.1080/1573062X.2016.1236133.

Ngo, K. T., Ernst, J. M., and Tokekar, P. 2018. 'Stacking Ensemble for auto ml Stacking Ensemble for auto ml'.

Nguyen, T., Huang, J. Z. and Nguyen, T. T. 2015. 'Two-level quantile regression forests for bias correction in range prediction.' *Machine Learning* 101(1–3): 325–343. doi: 10.1007/s10994-014-5452-1.

R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Ripley B., and Venables, W. 2016. Package 'nnet'. R version 7.3-12.

Romano, M., and Kapelan, Z. 2014. 'Adaptive water demand forecasting for near real-time management of smart water distribution systems.' *Environmental Modelling and Software* 60: 265–276. doi: 10.1016/j.envsoft.2014.06.016.

Scornet, E. 2017. 'Tuning parameters in random forests.' *ESAIM: Proceedings and Surveys* 60(2001): 144–162. doi: 10.1051/proc/201760144.

Shabani, S., Yousefi, P., Adamowski, J., and Naser, G. 2016. 'Intelligent Soft Computing Models in Water Demand Forecasting.' Water Stress in Plants. doi: 10.5772/63675.

Song, J. 2015. 'Bias corrections for Random Forest in regression using residual rotation.' *Journal of the Korean Statistical Society* 44(2): 321–326. doi:

10.1016/j.jkss.2015.01.003.

Stigler, S. M. 1997. ‘Regression towards the mean, historically considered.’ *Statistical Methods in Medical Research* 6(2): 103–114. doi: 10.1177/096228029700600202.

Tiwari, M., Adamowski, J., and Adamowski, K. 2016. ‘Water demand forecasting using extreme learning machines.’ *Journal of Water and Land Development* 28(1): 37–52. doi: 10.1515/jwld-2016-0004.

Touzani, S., Granderson, J. and Fernandes, S. 2018. ‘Gradient boosting machine for modeling the energy consumption of commercial buildings.’ *Energy and Buildings* 158: 1533–1543. doi: 10.1016/j.enbuild.2017.11.039.

Xenochristou, M., Blokker, M., Vertommen, I., Urbanus, J.F.X., and Kapelan, Z. 2018. CCWi2018: 032 Investigating the Influence of Weather on Water Consumption: a Dutch Case Study. Available from: https://ojs.library.queensu.ca/index.php/wdsa-ccw/article/view/12048/7605 [Accessed 23 January 2020].

Xenochristou, M. (2019). 'Water demand forecasting using machine learning on weather and smart metering data.' PhD thesis. University of Exeter. Retrieved from: https://ore.exeter.ac.uk/repository/handle/10871/39792 [Accessed 15 December 2019]

Xenochristou, M., Kapelan, Z., and Hutton, C. 2020. 'Using smart demand-metering data and customer characteristics to investigate the influence of weather on water consumption in the UK.' *Water Resources Planning and Management* 146(2). https://doi.org/10.1061/(ASCE)WR.1943-5452.0001148.

Xenochristou, M., Kapelan, Z., Hutton, C., and Hofman, J. 2020a. 'Water demand forecasting accuracy and influencing factors at different spatial scales using a Gradient Boosting Machine.' *Water Resources Research* (under revision).

Xenochristou, M., Kapelan, Z., Hutton, C., & Hofman, J. 2020b. 'Short-term forecasting of household water demand in the UK: An interpretable machine learning approach.' *J. Water Resources Planning and Management* (under revision).

Zhang, G., and Lu, Y. 2012. ‘Bias-corrected random forests in regression.’ *Journal of Applied Statistics* 39(1): 151–160. doi: 10.1080/02664763.2011.578621.

Zhang, P. 1993. ‘Model Selection Via Multifold Cross Validation.' *The Annals of Statistics* 21(1): 299-313.

Appendix

The ‘autoML’ module of ‘h2o’ trains a number of machine learning models (RF, XRT, GBM, DNN, and GLM) as well as tunes some of them (GLM, GBM, DNN) for the optimum set of hyperparameters. The model training stops according to a variety of stopping criteria. In this case, these were the stopping tolerance (0), stopping metric (MSE), and stopping rounds (1). This means that ‘h2o’ stops running when the MSE does not improve more than 0, over 2 consecutive iterations (for the same or different models). In addition, the maximum runtime was set to 2 hours, which means that the program stops running and saves the models developed up to this point, if none of the other criteria has been fulfilled. The same limits specified above were chosen for both groups of input parameters (Table 1), the one that includes past consumption, temporal, household, and weather characteristics, as well as the group that excludes past consumption. During this time, ‘h2o’ trained 335 models without past consumption and 147 models including past consumption data. Since additional variables add complexity to the model, they consequently increase training time, leading to less than half of models being trained within the same time frame.

Out of the six model types that were calibrated and tuned for the optimum set of hyperparameters, three of them (Random Forests - RFs, Extreme Gradient Boosting - XGBoost, Artificial Neural Networks - ANNs) were tuned using a pre-defined grid search space, whereas the Generalised Linear Model (GLM), Gradient Boosting Machine (GBM), and Deep Neural Network (DNN) were tuned automatically using a random grid search approach within the automated machine learning (‘autoML’) module of ‘h2o’ (h2o.ai, 2019b). The following section describes the number of hyperparameters used for tuning each model, as well as the final results of the search.

The implementations of the DRF and XRT models used to build the stacked model were the default model versions and are not described in the following.

Random Forest (RF)

The optimum (and default) value in regression for the number of random variables used for splitting (mtry) at each node is often considered to be the number of input variables divided by three. According to Table 1, the total number of variables is 14 for the models in Group 1, as opposed to 7 for the models in Group 2 (14 minus 7 days of past consumption). Therefore, the mtry was varied between 3 and 7 for the models with past consumption data and 2 to 4 without past consumption included as input. In both cases, the step size was set to 1. The number of trees was varied from 120 to 240, whereas the number of nodes from 20 to 120. In both cases, the step size was set to 20. The final values of mtry, ntree, and nodesize selected for each model (with and without past consumption) as a result of the grid search appear in Table A1.

Table A1: Hyperparameters for the RF model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
Mtry	6	7
Nodesize	100	40
Ntrees	160	200

Gradient Boosting Machine (GBM)

A total of nine hyperparameters were tuned by the automated machine learning module of 'h2o' for the GBM model. The final parameter values for each model, with and without past consumption data appear in Table A2. The 'auto' histogram type means that the cutting points tested for splitting are chosen by dividing the range of values of each variable in equal steps (20 steps here).

Table A2: Hyperparameters for the GBM model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
Ntrees	104	109
Max_depth	13	8
Learn_rate	0.05	0.05
Sample_rate	0.9	0.8
Col_sample_rate	0.4	0.4
Col_saple_rate_per_tree	0.4	1
Histogram_type	Auto	Auto
Min_split_imrpovement	1e-04	1e-05
Min_rows	10	15

Extreme Gradient Boosting (XGBoost)

The number of hyperparameters required for the XGBoost algorithm makes it difficult to define an extended search range for each parameter, due to the high dimensionality of the problem. Here, the XGBoost algorithm was tuned for seven input parameters. The search range for each parameter, as well as the selection of the subsample size was based on trial and error. The final values of each parameter appear in Table A3.

Table A3: Hyperparameters for the XGBoost model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
Nrounds	140	120
Max_depth	6	5
Colsample_bytree	0.4	0.7
Eta	0.05	1
Gamma	1	1
Min_child_weight	1.3	1.3
subsample	0.6	0.6

Artificial Neural Network (ANN)

The ANN implemented here has a single layer. The parameters used for tuning the model in this case were the size of the hidden layer and the decay, which where varied between 5-20 with a step of 1 and 0.01-0.1 with a step of 0.001, respectively. The above ranges for the grid space were chosen based on trial and error. The final values selected for each model are presented in Table A4.

Table A4: Hyperparameters for the ANN model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
Size	11	16
Decay	0.002	0.006

Deep Neural Network (DNN)

The DNN model was tuned for eight hyperparameters by the 'h2o autoML' platform. The hyperparameter values selected for the two DNNs, with and without past consumption data, are presented in Table A5.

Table A5: Hyperparameters for the DNN model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
epochs	270.4	131.2
Adaptive_rate	TRUE	TRUE
Activation	RectifierWithDropout	RectifierWithDropout
Rho	0.9	0.95
Epsilon	1e-08	1e-08
Input_dropout_ratio	0.2	0.1
Hidden	500	200 200 200
Hidden_dropout_ratios	0.4	0.2 0.2 0.2

Generalised Linear Model (GLM)

The GLM model was tuned for two hyperparameters, the alpha and missing values. The alphahyperparameter was varied between 0 and 1, with a step size of 0.2. The final values selected for each hyperparameter appear in Table A6. An alpha value of 0 indicates that a ridge regression (regularised linear regression) model is used to introduce penalties to the model building process, while MeanImputation means that the model replaces missing values with the mean.

Table A6: Hyperparameters for the GLM model with and without past consumption as input.

Hyperparameters	With past consumption	Without past consumption
Alpha	0	0
Missing values	MeanImputation	MeanImputation

More details regarding the model-building process as well as the hyperparameters available for tuning, their meaning, and the default hyperparameters of the models that were not mentioned here can be found in the online 'h2o' documentation (h2o.ai, 2019c).

All of the above hyperparameters are provided for reference only and for comparison purposes and do not replace the need to properly tune the above models based on the respective dataset.

Tables

Table 1: Input variables used to train the models

Variable Group	Model Input Variables	Group 1	Group 2
Past Consumption	1-7 days prior	X	
Temporal	Type of Day	X	X
	Season	X	X
Postcode	Area Postcode	X	X
Weather	Sunshine hours	X	X
	Air Temperature	X	X
	Humidity	X	X
	Days without rain	X	X
Total Variables		**14**	**7**

Table 2: Model comparison with (Group 1) and without (Group 2) past consumption, based on the test dataset. The highlighted in bold numbers represent the best accuracy achieved for each metric.

Model Groups	ID	Model Type	Bias Correction Method	MAPE (%) All days		MAPE (%) Peak days		R^2 (%)		MSE (l/postcode/day)	
				Train	Test	Train	Test	Train	Test	Train	Test
Group 1	1	RF	-	1.8	4.1	2.8	5.6	95.5	72.8	10	51
	2	XGBoost	-	3.0	4.2	4.5	6.0	86.3	72.5	27	53
	3	ANN	-	3.9	4.2	4.8	4.8	74.9	70.8	45	55
	4	GLM		4.1	4.2	5.8	5.8	71.3	70.6	51	55
	5	GBM	-	2.0	4.1	2.9	5.4	93.7	**74.1**	12	49
	6	DNN	-	3.5	4.2	4.7	5.2	79.7	72.5	36	51
	7	Stacked	-	2.2	**4.0**	3.2	5.1	91.8	**74.1**	15	**48**
	8	Stacked	BC1	2.2	**4.0**	2.8	4.8	91.4	**74.1**	16	**48**
	9	Stacked	BC2	2.6	**4.0**	3.3	**4.6**	88.7	**74.1**	20	**48**
	10	Stacked	BC3	2.2	**4.0**	3.0	5.1	91.6	**74.1**	15	**48**
	11	Stacked	BC4	2.2	**4.0**	2.9	4.8	91.5	**74.1**	15	**48**
Group 2	1	RF	-	2.3	4.6	3.5	6.0	92.2	68.0	16	60
	2	XGBoost	-	3.3	4.4	4.9	6.1	82.7	70.7	33	55
	3	ANN	-	4.3	4.7	5.9	6.0	68.5	65.1	56	65
	4	GLM		4.6	4.7	6.8	6.8	64.7	63.8	63	67
	5	GBM	-	3.1	**4.3**	4.2	5.6	84.0	70.9	29	54
	6	DNN	-	3.7	4.5	5.4	6.2	76.6	68.5	43	59
	7	Stacked	-	3.0	**4.3**	4.0	5.5	85.5	**71.1**	26	54
	8	Stacked	BC1	2.7	4.4	3.3	**5.1**	87.9	70.2	22	**51**
	9	Stacked	BC2	2.9	**4.3**	3.6	**5.1**	85.5	**71.1**	26	54
	10	Stacked	BC3	2.7	4.4	3.7	5.5	88.1	70.0	22	56
	11	Stacked	BC4	2.7	4.4	3.6	5.4	88.1	70.0	22	56

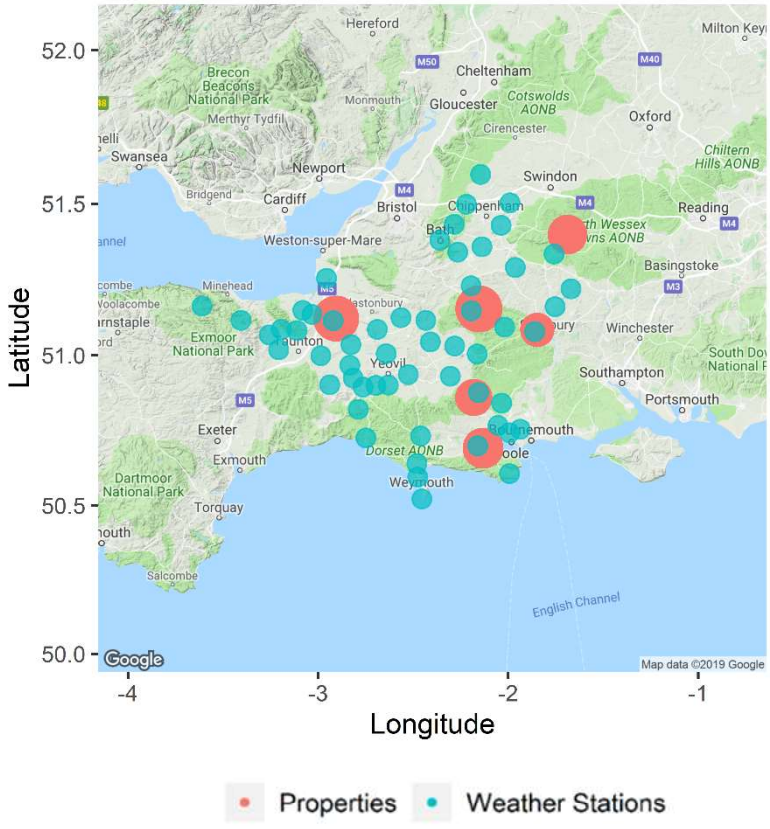

Figure 1: Location of area postcodes (red) and weather stations (blue).

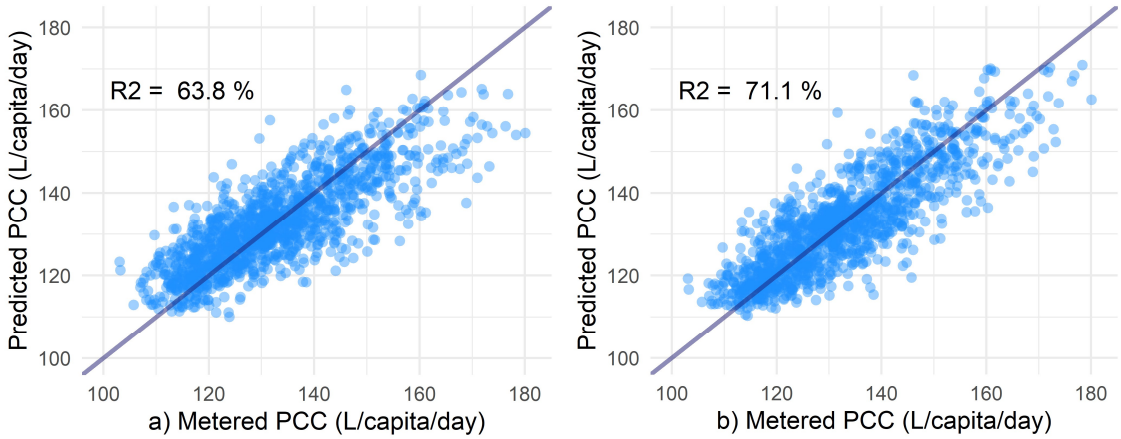

Figure 2: Metered VS predicted values for the GLM (plot a) and the stacked-BC2 model (plot b) when excluding past consumption data as input.