A Weight-adjusting Approach on an Ensemble of Classifiers for Time Series Forecasting

Lin Li
Department of Computer Science
Seattle University
Seattle, WA 98122
lil@seattleu.edu

Chun-Kit Ngan
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609
cngan@wpi.edu

ABSTRACT
In this study, we present a hybrid heterogeneous forecasting model that combines autoregressive integrated moving average (ARIMA) model and two machine learning classifiers (i.e. support vector machine (SVM) and artificial neural network (ANN)) for time series forecasting. The approach adjusts each model's weight based on their ability and history of predicting numerical values. A weighted numerical value based on each model's numerical output and their weight is calculated as the final output. An air quality dataset is used to evaluate our approach. We conduct the experimental comparison among our proposed weight-adjusting approach on the heterogeneous forecasting model, each individual model in the ensemble, and a hybrid homogenous forecasting model (e.g. random forest). It turns out that our proposed approach has a better performance than each single model in the ensemble and the hybrid homogeneous forecasting model in terms of mean absolute error (MAE) and mean absolute percentage error (MAPE).

CCS Concepts
Computing methodologies → Supervised learning by regression

Keywords
weight-adjusting; time series forecasting; ARIMA; SVM; ANN; ensemble of classifiers; random forests; MAE; MAPE

1. INTRODUCTION
Forecasting over univariate time series [1] is the most common and frequently used technique of predictive analytics since it has a significant impact on diverse private and public domains. For example, financial analysts use the historical indexes of foreign exchange rates (e.g. GBP/USD, EUR/USD, AUD/USD, etc.) to predict their future values to determine whether or not they should buy or sell a particular national currency [2]. Energy specialists on electric power microgrids use the past household energy consumptions to predict the future energy demand from which they can determine whether or not they should develop or construct more power plants and distributed lines to meet customers' energy use [3]. Environmental engineers investigate the gas composition collected from multi-sensor devices to predict the CO concentrations on population regions where the engineers can monitor the air quality variation over time [4]. The purpose of these value predictions over univariate time series is to assist the specialists in gaining the future insights on the problems within their particular domains. Once those significant indicators are accurately forecasted over univariate time series, those values can assist the professionals to make better decisions and take more reasonable actions promptly.

To support the reasonably accurate value predictions over univariate time series, scientists have proposed different forecasting models. Some common and popular models include autoregressive integrated moving average (ARIMA) [5, 6], artificial neural networks (ANN) [7, 8], and support vector machines (SVM) [9, 10]. An ARIMA model predicts a value over univariate time series as a linear combination of its own past values (i.e. autoregressive - AR), past errors (i.e. moving average - MA), and their predicted values replaced with the difference between the values and the previous values for eliminating the non-stationarity (i.e. integrated - I). This traditional model assumes that the underlying time series are generated from linear regression processes; hence, it may not be appropriate if the time series contain nonlinear data. To resolve the issue that an ARIMA model can only deal with time series of linear data, ANNs and SVMs play an important role on time series forecasting, as they are non-linear models. An ANN model consists of a large number of highly interconnected processing elements, i.e., artificial neurons, with nonlinear activation functions, e.g., sigmoid functions, working in parallel to conduct time series forecasting. An SVM model can perform linear classification, but it can also perform non-linear classification using kernel functions (e.g., Gaussian Radial Basis Function) by mapping their inputs into high-dimensional feature spaces. However, it is a challenging task for data specialists to select one of the above models to conduct time series forecasting due to diverse and unknown data variations (i.e. linear or nonlinear) in different domains. Data specialists always need to conduct the trial-and-error approach to select the best-fit model. In addition, choosing one of the models for time series forecasting may not be a practical approach because the real-world data may contain both linear and non-linear of data variations. To address this issue, hybrid forecasting models including homogeneous and heterogeneous are proposed and developed [11, 12, 13]. Random Forests (RFs), for example, is one of the common homogeneous classifiers that its model construction is based on the aggregation of a large number of decision trees. That is, the predictions of each individual tree are aggregated to produce a unique consensus prediction. However, there are some limitations on RFs in the regression analysis. First, the RF regression cannot predict beyond a range of training data because the predictions from RFs are done through averaging the results obtained in

ICISDM 2019, April 6–8, 2019, Houston, TX, USA

ACM ISBN 978-1-4503-6635-9/19/04...$15.00
https://doi.org/10.1145/3325917.3325920

several trees. The trees themselves output the mean value of the samples in each terminal leave node, so it is impossible for the average result to be outside the range of the training data, i.e. extrapolation (higher or lower than every sample), as the average result is always inside the range of its training samples. Second, its extreme values are often not predicted accurately either underestimating highs or overestimating lows. Third, although RF is a hybrid model, its construction is built based upon the same type of individual decision tree model that may not be able to capture and fit a wide variety of data distribution. To bridge the above gaps, heterogeneous models are constructed and developed. Heterogenous models combine multi-models using optimization techniques by assigning an optimal weight to each model based upon the specific characteristics of a dataset that results in a better performance (e.g. lower value of MAE and MAPE). However, learning the optimal weight for each model is a nonlinear optimization process that usually takes a long time to conduct partial differentiation and matrix manipulation to complete the weight computation. That is, the time complexity for learning the weights is at least quadratic [14], i.e. $O(mN^2)$, where m is the number of input models and N is the size of the validation dataset.

To address the above issues, we propose an approach that combines ARIMA, ANN, and SVM for time series forecasting. The proposed approach can adjust each model's weight based on their ability and history of predicting numerical value. A weighted numerical value based on each model's numerical output and their weight is calculated as the final output. An air quality dataset [15] is used to evaluate our approach. We conduct the pilot experimental comparison among the proposed weight-adjusting approach, each single model in the ensemble, and a hybrid homogenous forecasting model (e.g. random forest). It turns out that our proposed approach has a better performance in terms of mean absolute error (MAE) and mean absolute percentage error (MAPE). In addition, the time complexity of our proposed approach is *O(N)*, where N is the size of validation data set.

The rest of the paper is organized as follows. Section 2 describes our data used in the experimental study, the step-by-step process for learning the weights, and our algorithm for fine-tuning the weight of each classifier. In Section 3, we present and discuss our experimental result. Section 4 concludes the paper and discusses our future work.

2. DATA AND METHODS

2.1 Dataset

The air quality data set that we use in this work is from the machine learning repository [15] at the University of California, Irvine (UCI). The data set contains the responses of a gas multisensory device deployed on the field in an Italian city [15], including the data collected from March 2004 to April 2005. In total, there are 93578 instances of hourly averaged responses from five metal oxide chemical sensors. The sensors have collected CO concentration, relative humidity, Benzene concentration, etc. Missing values are tagged with -200.

In our experiment, we use the hourly averaged CO concentration (in mg/m^3), as a time series, from the air quality data set. First, we replace all the data instances that have -200 for CO concentration with 0 as it means missing data. We also remove all the data instances on 3/10/2004 (the 1st day of the recorded data) and all the data instances on 4/4/2005 (the last day of the recorded data) as the data of only 6 hours (18:00 – 23:00) are collected on 3/10/2004 and the data of only 15 hours (0:00 – 14:00) are collected on 4/4/2005. All other days have data collected every single hour (from 0:00 – 23:00). We then calculate the average CO concentration per each day through dividing the sum of all hourly averaged CO concentrations of the day by 24 hours. Finally, we use a 7-day sliding window to predict the average CO concentration per day based upon the data of previous seven days. That is to say, we use the average CO concentration in day 1, day 2, day 3, until day 7 as the input attributes, and predict the average CO concentration in day 8; similarly, we use the average CO concentration in day 2, day 3, day 4 until day 8 as the attributes, and predict the average CO concentration in day 9, so on and so forth. Since we want to use all the previous seven days to predict the average CO concentration of the next day, we do not conduct the feature selection. That means, each data record has seven attributes.

Table 1. Sample data from our experiment

A1	0.983333 (average on 08/01/2004)	1.282609 (average on 08/02/2004)	0.1 (average on 08/03/2004)
A2	1.282609 (average on 08/02/2004)	0.1 (average on 08/03/2004)	2.078947 (average on 08/04/2004)
A3	0.1 (average on 08/03/2004)	2.078947 (average on 08/04/2004)	1.241667 (average on 08/05/2004)
A4	2.078947 (average on 08/04/2004)	1.241667 (average on 08/05/2004)	1.075 (average on 08/06/2004)
A5	1.241667 (average on 08/05/2004)	1.075 (average on 08/06/2004)	1.486957 (average on 08/07/2004)
A6	1.075 (average on 08/06/2004)	1.486957 (average on 08/7/2004)	1.475 (average on 08/08/2004)
A7	1.486957 (average on 08/07/2004)	1.475 (average on 08/08/2004)	1.3 (average on 08/09/2004)
target	1.475 (average on 08/08/2004)	1.3 (average on 08/09/2004)	1.308696 (average on 08/10/2004)

Table 1 shows some sample data records in our experiment. In the table, all seven attributes (i.e. A1, A2, …, A7) are the average CO concentration of previous seven days of the target day. The last field (i.e. target) is the average CO concentration of the target day (please note that all the data in this table for the target day are the actual average values of the day collected by the sensors; they are not values predicted using our approach).

2.2 Learning Process

We use Weka to conduct regression using SVM and ANN. For SVM, SMOreg – the support vector machine for regression is used with the following settings: the complexity parameter C = 1.0, the training data is normalized, the polynomial kernel is used with an exponent value of 1.0, the learning algorithm proposed in [16] is used with a tolerance parameter 0.001 for checking stopping criterion. For ANN, the backpropagation is used with the following settings: 4 hidden layers, learning rate = 0.3, momentum = 0.2, the number of epochs to train = 500. For ARIMA, we use the two R libraries, including forecast [17] and tseries [18], to conduct the time series forecasting. First we create a time series

(TS) object using the ts() command from the tseries library. We then iterate ARIMA parameters (p, d, and q), i.e., the AR order, the degree of differencing, and the MA order, respectively, to create each TS model and conduct the time series forecasting using the forecast function to evaluate its performance respectively. We repeat the same process until we find the best-fitted ARIMA model that its Akaike information criterion (AIC) is the minimal.

The data set is partitioned into three disjoint sets. The training data set contains the data collected from March 2004 (3/11/2004) to July 2004 (7/31/2004), which is used to train the models. The validation data set (the data from 08/01/2004 to 11/30/2004) is used to adjust each classifier's weight in the ensemble when the classifiers are used all together to conduct the time series forecasting. The details are discussed in Section 2.3. The data from 12/1/2004 to 04/03/2005 is used as the testing data to evaluate the performance of the ensemble of classifiers. The procedure of training, validation, and testing are shown in Fig. 1. The procedure is similar to the procedure that we used in our previous work [19] to use an ensemble of classifiers for classification. Since we perform time series forecasting in this work, we do not need to conduct the voting proposed in [19] at the end of phase 3 (use testing data to test the performance of weight-adjusting approach) to combine the decisions of multiple classifiers in the ensemble. Instead, we only need to calculate the weighted predicted value (as discussed in Section 2.3).

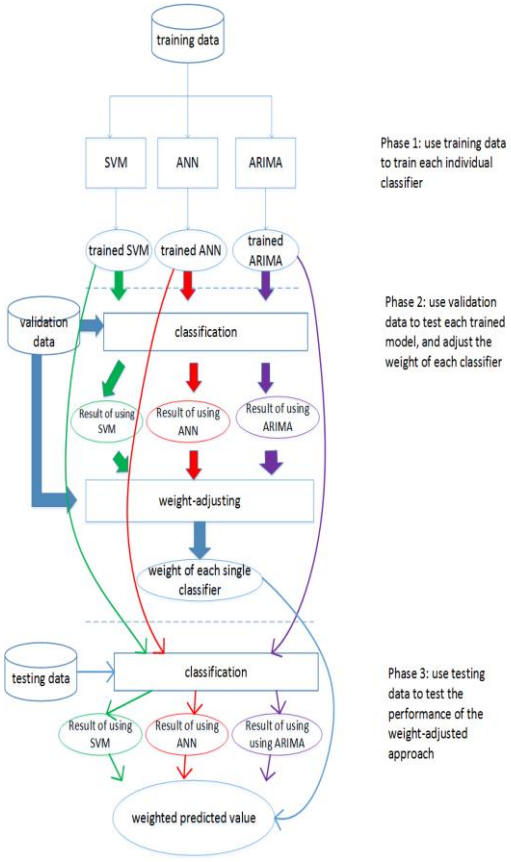

Figure 1 The procedure of training, validation, and testing in our experiment

2.3 Our Weight-Adjusting Approach

The proposed weight-adjusting approach is based on an ensemble of three classifiers. For a given instance, for each classifier, the algorithm calculates the distance (i.e. difference) between the predicted value and the actual value. Then it identifies the classifier(s) which have the closest distance. For example, if classifier A has a closer distance than the other classifiers, say B and C, then classifier A's weight is increased and classifier B's and C's weights are decreased. This is based on the consideration that a more powerful classifier should be able to predict a value that is

ALGORITHM 1: Adjusting the Weight of Classifier.

Input:
- *N*: the number of instances in the validation dataset
- Array *A* of size *N* which saves the prediction results of validation dataset using classifier A
- Array *B* of size *N* which saves the prediction results of validation dataset using classifier B
- Array *C* of size *N* which saves the prediction results of validation dataset using classifier C
- Array *L* of size *N* which saves the actual value of each instance in the validation dataset
- *step*: the step size used to change weights

Output:
- *W*: a vector {W_A, W_B, W_C} that stores the weight of classifier A, B, and C respectively

1 $W_A = W_B = W_C := \frac{1}{3}$
2 **for** i := 0 **to** $N - 1$ **do**
3 distA = |A[i] -L[i]|
distB = |B[i] -L[i]|
distC = |C[i] -L[i]|
totalDistance = distA + distB + distC;
4 **if** there is only one classifier that has the smallest distance (e.g. distA < distB and distA < distC)
5 twoDistance = the sum of distances using other two classifiers (e.g. distB + distC)
6 weight of the smallest distance classifier += step * twoDistance/totalDistance
(e.g. weight of classifier A += step * twoDistance/totalDistance;)
7 For each of the other two classifiers that have larger distance
weight of the classifier -= step * the distance of this classifier/totalDistance
(e.g. weight of classifier B -= step * distB/totalDistance;
weight of classifier C -= step * distC/totalDistance;)
8 **if** there are two classifiers that have the same distance and it is smaller than the 3rd classifier (e.g. distA = distB < distC)
9 weight of the two classifiers that have smaller distance += step
(e.g. weight of classifier A += step;
weight of classifier B += step;)
10 weight of the classifier that has the largest distance -= 2 * step
(e.g. weight of classifier C -= 2 * step;)
11 **return** *W*

closer to the actual value. A classifier should get a credit (i.e. the weight should be increased) if its predicted value is closer to the actual value. If the predicted value is far away from the actual value, the classifier should be penalized (i.e. the weight should be decreased). How large the weight is increased or decreased is proportional to how close or how far away the predicted value is from the actual value. If there are two classifiers (e.g. A and B) that have the same distance and they are both closer than the third classifier (e.g. C), then the weight of both classifiers A and B is increased and the weight of classifier C is decreased. If all three classifiers have the same distance, then there is no weight adjustment. The algorithm for adjusting the weight is described in Algorithm 1. As shown in Algorithm 1, the weight of each classifier in the ensemble is adjusted in one pass of visiting each data record in the validation data set. Hence, the time complexity of the proposed approach is linear - *O(N)*, where N is the size of the validation dataset.

After the weight of the three classifiers is adjusted based on the validation dataset, the final weight of each classifier is used on the testing dataset to calculate the weighted predicted value. For example, if the weight of classifier A is 0.3, the weight of classifier B is 0.5, the weight of classifier C is 0.2, and the predicted value using classifier A is 0.45, the predicted value using classifier B is 0.39, the predicted value using classifier C is 0.56, then the final predicted value using the ensemble of three classifiers is 0.3 * 0.45 + 0.5 * 0.39 + 0.2 * 0.56 = 0.442.

3. RESULTS AND DISCUSSION

In our experimental study, we use SVM, ANN, and ARIMA in the ensemble of classifiers. We evaluate our proposed approach on the ensemble, and each single classifier in the ensemble, in terms of MAE and MAPE, as shown in Table 2. We also take Random Forests as an example of a hybrid homogenous forecasting model and compare its MAE and MAPE values with our proposed approach (shown in Table 2). Weka is used to conduct regression using Random Forest with its default settings. The comparison shows that our proposed weight-adjusting approach can lead to a smaller MAE value and a smaller MAPE value than using each single classifier in the ensemble and the Random Forests. We also notice that although the MAE and MAPE values of RFs are smaller than that of ANN and ARIMA, they are larger than the MAE and MAPE values of SVM. This means that RFs may not be able to fit a wide variety of data distribution because its construction is built upon the same type of decision tree model.

Table 2. Comparison of MAE and MAPE on the testing dataset using each single classifier, random forests, and the ensemble of three classifiers using our proposed approach

Classifier	MAE	MAPE
SVM	0.5962	32.41%
ANN	0.6705	34.94%
ARIMA	0.7113	42.04%
Random Forest	0.6283	36.17%
The ensemble of using our proposed weight-adjusting approach	**0.5779**	**30.52%**

Below are the formulas of MAE and MAPE. MAE is a common metric to measure accuracy for continuous variables. MAPE is a measure of prediction error in forecasting. The formulas are shown below.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|A_i - F_i| \quad (1)$$

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{A_i - F_i}{A_i}\right| \quad (2)$$

In the formulas, N is the number of data points in the time series, A_i and F_i is the actual value and the forecast value of the data point i.

The experimental result is affected by the value of *step* in the algorithm and how the classifiers change weight. For the value of *step*, we test all values of *step* in a distance of every 0.00001, starting from 0.0001 to 0.1 (i.e. 0.0001, 0.00011, 0.00012, etc.). We also try different ways of adjusting weight. It turns out the current way of adjusting weight and using the step size of 0.00106 can provide the best result. Fig. 2 presents how the value of step size influences the result of MAE and MAPE, using a small set of step sizes (close to the step size that leads to the lowest point of MAE and MAPE) that we have tried. As shown in the figure, when the step size goes up approaching around 0.001, the MAE value and MAPE values go down. After the MAE and MAPE values reach the lowest point, they keep increasing as the step size increases.

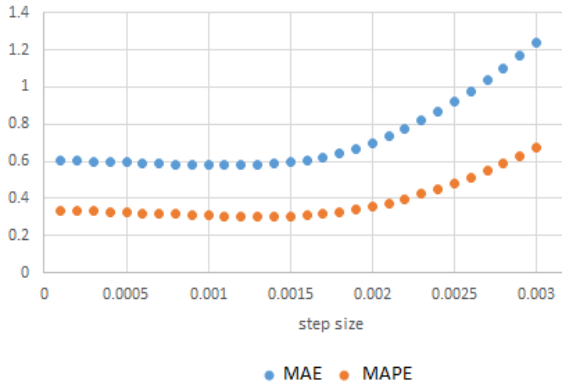

Figure 2 The plot of MAE value, MAPE value and the step size

Our experiment allows the weight of a classifier to be negative, which means that the classifier contributes negatively to the result. In our experiment, the initial weight of each classifier is 0.3333. After the weight adjusting, the weight of SVM, ANN and ARIMA is 0.6170, 0.4544 and -0.0716, respectively. By observing the experimental result, we find out that the weight of SVM is always larger than that of the other two classifiers, and the weight of ARIMA is always the smallest no matter what the step size is used. The weight of ARIMA can even reach a negative value. The result indicates that SVM plays a more important role in the ensemble for time series forecasting. It also indicates that ARIMA might not be a good choice for time series forecasting on the air quality dataset, which is non-linear.

We also evaluate the performance of combining any two classifiers in the ensemble of SVM, ANN, and ARIMA. The weight-adjusting approach is similar to the one that is proposed in Algorithm 1, except that - (1) there are only two classifiers; (2) the initial weight of each classifier is 0.5; (3) how the weight of each classifier is changed is different. If one classifier (e.g. classifier A) has a

smaller distance (explained in Algorithm 1), then the weight of classifier A is increased by step * distB/ (distA + distB), and the weight of classifier B is decreased by step * distB/(distA + distB). There is no weight change if two classifiers have the same distance. Table 3 presents the MAE and MAPE values of the combination of any two classifiers, as well as the weight of each classifier and the step size used in each combination.

Table 3 Performance of the combination of two classifiers on the testing dataset

Classifier A (weight)	Classifier B (weight)	step	MAE	MAPE
ANN (0.3742)	SVM (0.6258)	0.00101	0.5777	0.3087
SVM (1.1025)	ARIMA (-0.1025)	0.0009	0.5955	0.3204
ANN (0.6799)	ARIMA (0.3201)	0.00074	0.6497	0.3521

It can be seen from Table 3 that overall the combination of two classifiers has a better performance than each single classifier in the combination, except that the combination of ANN and ARIMA has a slightly higher MAPE value than that of ANN. The combination of ANN and SVM has a slightly smaller MAE value than using all three classifiers (0.5779 as shown in Table 2), but the MAPE value is slightly higher than using the ensemble of three classifiers.

4. CONCLUSION AND FUTURE WORK

In this paper, we propose a weight-adjusting approach for time series forecasting based on the prediction results of an ensemble of three classifiers – SVM, ANN and ARIMA. Our work has three contributions. First, our approach has a better performance than each single classifier in the ensemble and a homogenous forecasting model (RFs) in terms of MAE and MAPE. The combination of any two classifiers in the ensemble also has a better performance than each single classifier in the combination. Second, the ensemble of classifiers combines linear models (ARIMA) and non-linear models (ANN and SVM). Third, the time complexity of the proposed approach is linear (i.e., $O(N)$) instead of quadratic (i.e., $O(mN^2)$), where m is the number of input models and N is the size of the validation dataset. Data specialists can use the approach to conduct time series forecasting which aims to take multiple classifiers' decision into consideration.

In the future, we will extend the approach to work with an ensemble of more than three classifiers. It is also critical to explore how to perform the classifier selection to remove classifiers that contribute negatively in the ensemble to the time series forecasting result. In addition, more time series will be used in the experiment to evaluate the proposed approach.

5. REFERENCES

[1] Kumar, U.D. 2017. Business Analytics, Wiley India Pvt. Ltd., Daryaganj, New Delhi, India.

[2] Investing.com. 2018. Real Time Technical Analysis Summary. Foreign Exchange Market, https://www.investing.com/technical/technical-summary (Accessed December 22, 2018).

[3] Gajowniczek, K., & Ząbkowski, T. 2017. Electricity forecasting on the individual household level enhanced based on activity patterns. PloS one, 12(4), e0174098. doi:10.1371/journal.pone.0174098.

[4] Ghadiri, M., Marjani, A., & Shirazian, S. 2017. Development of a mechanistic model for prediction of CO_2 capture from gas mixtures by amine solutions in porous membranes. Environmental Science and Pollution Research, 2017, Volume 24, Number 16, Page 14508.

[5] Fuller, W. 1979. Introduction to Statistical Time Series, John Wiley and Sons, New York, USA.

[6] Yaffee, R.A., and McGee, M. 2000. An Introduction to Time Series Analysis and Forecasting: With Application and SPSS, Academic Press, New York, USA.

[7] Hassoun, M. 1995. Fundamentals of Artificial Neural Networks, The MIT Press, Cambridge, MA, USA.

[8] Heaton, J. 2015. Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks, Heaton Research, Inc, Chesterfield, MO, USA.

[9] Adankon, M.M., and Cheriet, M. 2009. Support Vector Machine. Encyclopedia of Biometrics, Springer, Boston, MA, USA.

[10] Suthaharan, S. 2016. Support Vector Machine. In: Machine Learning Models and Algorithms for Big Data Classification. *Integrated Series in Information Systems*, vol 36. Springer, Boston, MA, USA.

[11] Zhang, G.P. 2003. Time series forecasting using a hybrid ARIMA and neural network model, *Neurocomputing*, Volume 50, January 2003, 159-175. Elsevier Science B.V., Amsterdam, The Netherlands.

[12] Khashei, M., and Bijari, M. 2010. An artificial neural network (p, d, q) model for timeseries forecasting, *Expert Systems with Applications*, Volume 37, Issue 1, January 2010, 479-489. Elsevier Science B.V., Amsterdam, The Netherlands.

[13] Petrakova, A., Affenzeller, M., & Merkuryeva, G. 2015. Heterogeneous versus Homogeneous Machine Learning Ensembles. *Information Technology and Management Science*, Volume 18, Issue 1, Februray 2010, 135–140. The Journal of Riga Technical University.

[14] Hochbaum, D.S. 2007. Complexity and Algorithms for Nonlinear Optimization Problems. Annals of Operations Research, Volume 153, Issue 1, September 2007, 257–296. Springer Nature Switzerland AG.

[15] Air Quality Data Set, [Online], Available: https://archive.ics.uci.edu/ml/datasets/Air+Quality

[16] Shevade, Shirish Krishnaj, et al, Improvements to the SMO algorithm for SVM regression, *IEEE transactions on neural networks* 11.5 (2000): 1188-1193.

[17] Rob Hyndman, R., et al. 2017. forecast: Forecasting Functions for Time Series and Linear Models. https://cran.r-project.org/web/packages/forecast/index.html.

[18] Trapletti, A., et al. 2018. tseries: Time Series Analysis and Computational Finance. https://cran.r-project.org/web/packages/tseries/index.html.

[19] Lin Li, Ben C.K. Ngan. 2017. A Weight-Adjusted-Voting Framework on an Ensemble of Classifiers for Improving Sensitivity, *Intelligent Data Analysis (IDA)*, vol.21, no.6 (2017), 1339-1350.