An Evaluation of Regression Algorithms Performance for the Chemical Process of Naphthalene Sublimation

Florin Leon, Andrei-Ștefan Lupu, Sabina-Adriana Floria, Doina Logofătu, Silvia Curteanu

An Evaluation of Regression Algorithms Performance for the Chemical Process of Naphthalene Sublimation

Silvia Curteanu[1], Florin Leon[2], Andrei-Ștefan Lupu[3], Sabina-Adriana Floria[4], Doina Logofătu[5]

[1]Department of Chemical Engineering and Environmental Protection
"Gheorghe Asachi" Technical University of Iaşi
Iași, Romania
scurtean@ch.tuiasi.ro

[2,4]Department of Computer Science and Engineering
"Gheorghe Asachi" Technical University of Iaşi
Iași, Romania
florin.leon@tuiasi.ro, sabina.floria@tuiasi.ro

[3]School of Electronics and Computer Science
University of Southampton, UK
Southampton, UK
asl1u12@soton.ac.uk

[5]Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt, Germany
logofatu@fb2.fra-uas.de

Abstract. Different regression algorithms are applied for predicting the sublimation rate of naphthalene in various working conditions: time, temperature, trainer rate and shape of the sample. The original Large Margin Nearest Neighbor Regression (LMNNR) algorithm is applied and its performance is compared to other well-established regression algorithms, such as support vector regression, multilayer perceptron neural networks, classical k-nearest neighbor, random forest, and others. The experimental results obtained show that the LMNNR algorithm provides better results than the other regression algorithms.

Keywords: regression, large margin, nearest neighbor, naphthalene sublimation.

1 Introduction

Machine learning is a subdomain of artificial intelligence whose popularity and success are constantly growing [1, 2]. Its main goal is to extract high-level patterns, i.e. knowledge, from large amounts of raw information, patterns that can provide more abstract and useful insight into the data under study. Many problems in science and social science can be expressed as classification or regression problems, where

one does not know an analytical model of some underlying phenomenon, but sampled data is available through experiments or observations, and the aim is to define a predictive model based on those samples. To date, many such algorithms have been proposed, which belong to different paradigms, e.g. neural networks, nearest neighbor, decision trees, support vector machines, Bayesian approaches, etc.

Unfortunately, there is no single best algorithm that can handle the large variety of situations encountered in practice. Each method has its own advantages and disadvantages. They are mainly related to the flexibility or complexity of the models and their generalization capabilities. For a non-trivial pattern, using a very simple model may result in poor performance, whereas using an overly complex model can result in overfitting, i.e. very good results for the training set and poor results for the test set or prediction, in general. Therefore, one must make several choices when dealing with such a problem: first, to establish the most appropriate learning method and, second, to control the complexity of the model generated with that learning method by changing its specific parameters.

In the present paper, we investigate the performance of some well-established algorithms in comparison to an original regression algorithm, namely the *Large Margin Nearest Neighbor Regression* (LMNNR), which combines the idea of nearest neighbors with that of a large separation margin, typical of support vector machines. The sublimation of naphthalene was chosen to illustrate these methodologies based on the difficulties involved due to the toxicity of the process, in which case predictions on the model become recommended and useful.

We organize our paper as follows. Section II presents a selection of related work about regression algorithms applied for the modeling of chemical processes. Section III describes the dataset used for the experiments and Section IV presents the algorithms employed to model it. Section V describes some experimental results, while Section VI contains the conclusions of our work.

2 Related Work

There are many applications of artificial intelligence and soft computing methods in the domain of chemical engineering, especially for modeling and optimization. In this section, we review several applications of regression algorithms for chemical processes.

Article [3] proposes a combination of online support vector regression with an ensemble learning system to adapt to nonlinear and time-varying changes in process characteristics and various process states in a chemical plant. [4] uses a probabilistic combination of local independent component regression in order to assess the quality of chemical processes with multiple operation modes. [5] addresses a non-linear, time-variant problem of soft sensor modeling for process quality prediction using locally weighted kernel principal component regression. [6] uses multiple linear regressions and least squares support vector regression to model and optimize the dependency of methyl orange removal with various adsorption influential parameters. [7] compares the performance of support vector regression, neural network and random forest models in predicting and mapping soil organic carbon stocks.

In [8], the authors make a thorough presentation of neural networks used for bioprocessing and chemical engineering, with applications in process forecasting, modeling, control of time-dependent systems, and the hybridization between neural networks and expert systems.

The issue of predicting sublimation thermodynamics, such as enthalpy, entropy, and free energy of sublimation using machine learning methods was addressed in [9]. Semi-empirical models were used to model systems of solids and supercritical fluids in order to determine sublimation pressures and sublimation enthalpies, and then to model different multiphasic equilibriums [10].

Some of the recent research of the authors of the present paper addressed a performance comparison of different regression methods for a polymerization process with adaptive sampling [11], a comparison between simulation and experiments for phase equilibrium and physical properties of aqueous mixtures [12], an experimental analysis and mathematical prediction of cadmium removal by biosorption [13] and the prediction of corrosion resistance of some dental metallic materials with an original adaptive regression model based on the k-nearest-neighbor regression technique [14].

3 The Naphthalene Sublimation Dataset

Our case study is naphthalene sublimation – a physical process of solids that transition directly into vapors. This technique is one of the most convenient methods to study heat and mass transfer. In addition, the rate of sublimation, the amount of solid converted to vapor per time unit and solid area unit is used to study problems related to environment protection, health protection, transportation safety and security, meteorology, by determining the concentration of various substances in the environment and the dynamical properties in a wind tunnel.

In a previous approach [15], a series of experiments were performed to investigate the sublimation of the naphthalene samples under atmospheric pressure in air as entrainer, without recycle. Our experimental data fulfill a necessary condition for empirical modeling: a sufficient number of data was obtained which uniformly cover the investigated domain.

The sample weight was measured continuously as a function of time, at different air flow characteristics. The experimental data is then used to calculate the mass transfer rate, the degree of sublimation, the sublimation front position; the influence of air flow characteristics was also evaluated.

More details on experiments and data processing can be found in [15], where neural network modeling was performed. In the current work, a more efficient algorithm, LMNNR, was applied, comparatively with other algorithms: linear regression, support vector regression, neural networks, k-Nearest Neighbors, K*, and Random Forest. In addition, a large dataset was used here (1323 instances) including different shapes of the samples, while in [15] only spherical samples were considered (150 instances).

The data gathered from experiments contains four variables as inputs: the shape of the sample (i.e. pallets, small pills, large pills and rods), time, air speed (the trainer) and temperature, and one output: the speed of naphthalene sublimation.

Consequently, the modeling purpose was to evaluate the performance of the process, quantified by the sublimation rate depending on process time, entrainer temperature, and entrainer flow rate.

In order to apply the instance-based methods, the data is normalized between 0 and 1, independently for each numerical attribute.

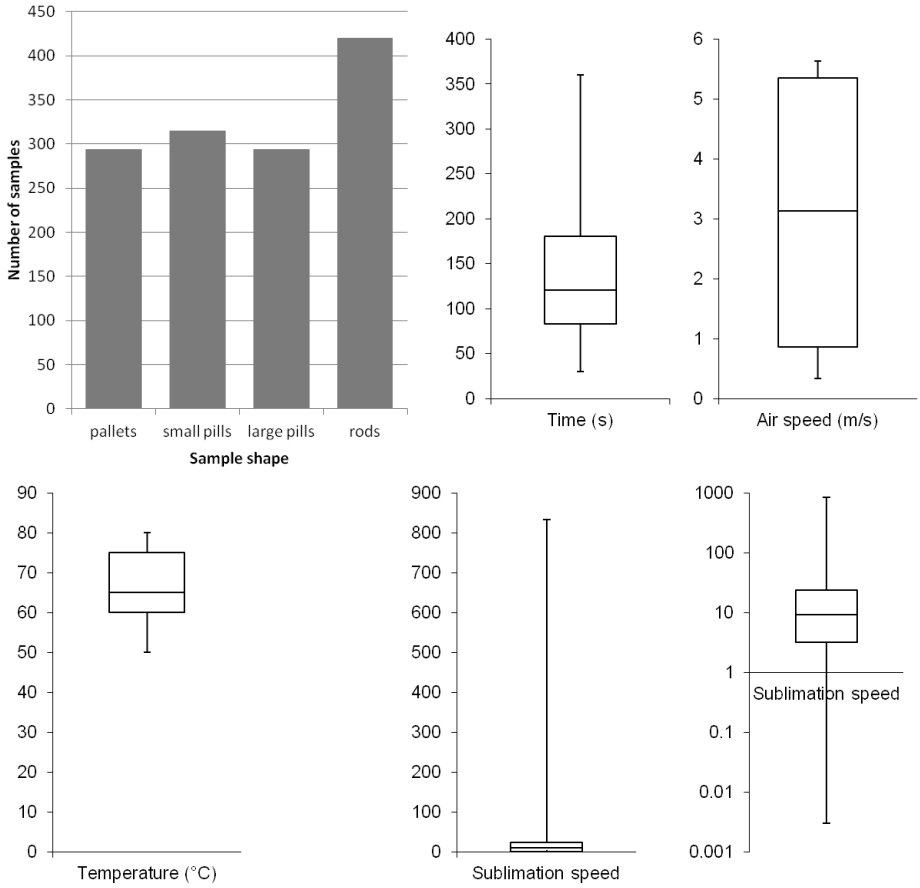

Fig. 1. Statistics of the inputs and the output of the naphthalene sublimation dataset

Fig. 1 presents some statistics regarding the distribution of the data before normalization: the histogram for the first discrete input and a box plot for each numerical input, showing the minimum value, the first quartile, the median, the third quartile and the maximum value. For the output, two box plots are included, with a linear and a logarithmic scale. The output has values between 0.003 and 832.98, with the mean of 34.95 and the median of 9.24. There are a few greater values far from the median, but they are not outliers; they are important results of the process, difficult to learn, and which need to be handled accordingly by the regression models.

4 Regression Algorithms

The goal of the paper was to find a good model for the naphthalene sublimation data. The first step was to apply classical methods, with known good performance, implemented in *Weka* [16]. This was intended to constitute a basis for comparison with the original LMNNR algorithm. From the large number of algorithms in Weka, a few were selected which, in previous studies, were noticed to yield good performance for a large number of regression problems. Thus, neural networks, support vector machines, nearest neighbor, K-Star and random forest were selected. The details about their structure and operation are given below.

It must be emphasized that these techniques have very different nature and assumptions, and, by comparing the LMNNR results with the best results obtained with either of these classical algorithms, we can underline that the algorithm proposed by the authors is, in fact, a good alternative for regression.

4.1 Classical Algorithms

Neural networks in the form of *multilayer perceptrons* (MLP) are often used in classification and regression problems. The structure of an MLP contains an input layer, an output layer and one or more hidden layers of neurons. Each neuron sums the weighted input data of the neurons in the previous layer, to which another term (bias) is added, and the result is sent to the neurons in the next layer through a nonlinear transformation called an activation function. Each connection has an associated weight. In the training process, the weights and biases are adjusted such that the output of the network should match the desired output of the vectors from the training set. The training algorithm used most often is back-propagation [17]. It aims to minimize the mean-squared error between the desired output and the computed one using the gradient descent method.

The *Epsilon-Support Vector Regression* (ε-SVR) algorithm tries to approximate the desired continuous output within a tolerating error ε while using the idea of the large margin characteristic of support vector machines [18]. When the data is not linearly separable, the ε-SVR algorithm uses kernels to transform them into a higher-dimensional space. There are several types of functions that can be used as kernels, e.g. polynomial or radial basis functions (RBF). If some training instances still do not satisfy the constraints, slack variables are introduced to allow some errors (soft margin). The number of these erroneous instances can be controlled with a cost parameter C. If the value of C is decreased, a larger number of incorrectly classified training instances is allowed, which can however lead to better generalization.

The *k-Nearest Neighbor* (kNN) algorithm is based on the choice of k nearest neighbors using a distance function as a criterion and the output is computed by aggregating the outputs of those k training instances. As a distance function, one can use Euclidian or Manhattan distance, usually particularizations of the Minkowski distance. Choosing the value of k is important. If k is too small, then the classification can be affected by the noise in the training data, and if the value of k is too large, then distant neighbors can affect the correctness of the results. To avoid the difficulty of finding an optimum value for k, one can weight the neighbor influence. The neighbors

have a greater weight as they are closer to the instance, while those farther apart have a smaller weight.

The *K-Star* algorithm [19] is an instance-based classifier that very much resembles the k-Nearest Neighbor algorithm presented before. Its novelty comes from the usage of an entropy metric in its similarity function, rather than the usual distance metric. It has been shown in the literature that such an approach has beneficial outcomes for certain industry-related problems [20]. The K-Star algorithm can also be used for regression purposes, similarly to how k-Nearest Neighbor is used.

A *random forest* [21] is composed of a collection of classification or regression trees. Each tree is generated using random split tests on slightly different training set generated using bagging. The output of a new instance is computed by aggregating the outputs of the individual trees.

4.2 The Large Margin Nearest Neighbor Regression Algorithm

The performance of the above algorithms was compared to that of an original algorithm, *Large Margin Nearest Neighbor Regression* (LMNNR) [22, 23].

The support vector machines, in a classification context, rely on the idea of finding a large margin between classes by solving an optimization problem. This idea was used in conjunction with the k-Nearest Neighbor method, also for classification [24]. Its main assumption is to change the distance metric of the kNN space by using a matrix:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i - \mathbf{x}_j\right)^T \mathbf{M}\left(\mathbf{x}_i - \mathbf{x}_j\right). \tag{1}$$

If **M** is a diagonal matrix, the weights of the neighbors are:

$$w_{d_M}(\mathbf{x}, \mathbf{x}') = \frac{1}{d_M(\mathbf{x}, \mathbf{x}')} = \frac{1}{\sum_{i=1}^{n} m_{ii} \cdot \left(x_i - x'_i\right)^2}. \tag{2}$$

Equation (2) involves a single, global matrix **M** for all the instances. However, it is possible to have different distance metrics for the different instances or groups of instances. Thus, *prototypes* can be used which are defined as special locations in the input space of the problem, and each prototype P has its own matrix $\mathbf{M}^P$. When computing the distance weight to a new point, an instance uses the weights of its nearest prototype, i.e. m_{ii}^P instead of m_{ii} in equation (2).

Finding the appropriate matrices is achieved by solving an optimization problem. In a simplified formulation, the objective function F, which is to be minimized, takes into account two criteria with equal weights, F_1 and F_2, described below. In order to briefly explain the expressions of these functions, the following notations were made, where d_M means the weighted square distance function using the weights we search for: $d_{ij} = d_M(x_i, x_j)$, $d_{ik} = d_M(x_i, x_k)$, $g_{ij} = |f(x_i) - f(x_j)|$, $g_{ik} = |f(x_i) - f(x_k)|$.

The first criterion is:

$$F_1 = \sum_{i=1}^{n} \sum_{j \in N(i)} d_{ij} \cdot (1 - g_{ij}), \tag{3}$$

where $N(i)$ is the set of the nearest k neighbors of instance i, e.g. $k = 3$. Basically, this criterion says that the nearest neighbors of i should have similar values to the one of i, and more distant ones should have different values.

The second criterion is expressed as follows:

$$F_2 = \sum_{i=1}^{n} \sum_{j \in N(i)} \sum_{l \in N(i)} \max(1 + d_{ij} \cdot (1 - g_{ij}) - d_{ik} \cdot (1 - g_{il}), 0). \tag{4}$$

Here, the distance to the neighbors with close values (the positive term) is minimized, while simultaneously trying to maximize the distance to the neighbors with distant values (the negative term). An arbitrary margin of at least 1 should be present between an instance with a close value and another with a distant value.

For optimization, both an evolutionary algorithm and an approximate differential method following the central difference definition of the derivative can be used.

The estimated output of a new query instance $\mathbf{x}_q$ is computed as follows. Its k nearest neighbors are identified using the distance metric from equation (1). The weights of these neighbors are computed with equation (2) and then normalized:

$$w_{d_M}^n(\mathbf{x}_i, \mathbf{x}_q) = \frac{w_{d_M}(\mathbf{x}_i, \mathbf{x}_q)}{\sum_{j=1}^{k} w_{d_M}(\mathbf{x}_j, \mathbf{x}_q)}. \tag{5}$$

Finally, the output is computed as a weighted average of the neighbor outputs:

$$\tilde{f}(\mathbf{x}_q) = \sum_{i=1}^{k} w_{d_M}^n(\mathbf{x}_i, \mathbf{x}_q) \cdot f(\mathbf{x}_i). \tag{6}$$

5 Results and Discussion

In this section, the choice of parameters for different regression methods is explained. For each algorithm, multiple experiments with different parameter values were performed. The tables containing the results only display those with the best performance in terms of correlation coefficient (r) and root mean square error (*RMSE*).

In order to compare the performance of the various algorithms, the cross-validation method with 10 folds was used. Also, since an objective comparison was intended, the data set was randomly divided into 10 groups (iteratively one for test and the rest for training) and the same groups were used by all the algorithms. It was considered that this methodology is particularly important to compare the algorithms implemented in Weka with the original implementation of the LMNNR algorithm.

The results obtained for individual test groups, although interesting, were omitted in the results section, and only the aggregated results are displayed in the following tables.

5.1 Parameters of Regression Methods

Multilayer Perceptron Neural Network (MLP). For the problem at hand, repeated experiments showed that a neural network produces best results when given a low learning rate. The momentum parameter also has a great impact on learning. Its optimal value tends to be around 0.4 or 0.5. The number of hidden layers was automatically chosen by Weka. This option yielded the best outcomes because the optimal number of hidden layers tends to vary between cross-validation sets, making it hard to achieve similar performance with manually chosen values. The encoding for the discrete input is "one-hot", leading to 7 inputs and 1 output. The best network architecture was the one with one hidden layer containing 4 neurons with sigmoid activation functions, and with the output neuron with a linear activation function. 1000 epochs for training were found to be an acceptable compromise between the quality of the resulting model and the overall training time.

Support Vector Regression (SVR). The Epsilon-SVR algorithm achieved a very good overall fit if the kernel used was based on radial basis functions. The kernel type choice was vastly influential on the outcome. RBF, therefore, yielded a correlation that was at least 20% better than all of the other options (linear, polynomial and sigmoid). The best results were obtained with relatively large values of the parameters: $\gamma = 14$, $C = 10$, whereas ε was best kept at a low value, i.e. $\varepsilon = 0.001$. Fine-tuning these parameters helped improve the algorithm performance significantly, such that the final correlation was the best out of all the algorithms tested with Weka.

***k*-Nearest Neighbor (*k*-NN).** The optimal number of neighbors used in this algorithm is in this case 2. The correlation dropped significantly if the number of neighbors was increased above this value. The search method used was the linear nearest neighbor search. A slight improvement was achieved by using the Manhattan distance as metric, instead of the Euclidean distance.

K-Star (K^*). The only numeric parameter that this algorithm takes, the global blending index, was optimal at low values. In the experiments, the value 3 was used. The parameter, however, influenced the outcome in a slight manner (~5% correlation improvement). The entropic auto blend functionality provided by Weka was turned off for these experiments.

Random Forest (RF). In the case of this algorithm, the number of trees parameter plays an important role in the overall performance. Several tests were conducted to determine the optimal value of this parameter and the best outcome was recorded with a value of approximately 200 trees. Although the difference in performance obtained by optimizing this parameter was only around 3%, it allowed Random Forest algorithm to yield one of the best correlations for the data.

Large Margin Nearest Neighbor Regression (LMNNR). For this algorithm, the parameters are the number of prototypes, the number of optimization neighbors and the number of regression neighbors. Different combinations of values for these parameters were attempted. Since the LMNNR results are not deterministic, because

the initialization of the matrices is random and then optimized, the best results were included out of 100 algorithm runs for each configuration.

5.2 A Comparison Between Algorithm Performance

In Tables 1 and 2, one can see the best results achieved with the use of the regression algorithms presented in the previous section.

Table 1. The best results obtained for optimized configurations by algorithms in Weka

Algorithm	**Parameters**	***r***	**RMSE**
ε-SVR	C: 10; ε: 0.001; γ: 14; kernel: RBF	0.91514	0.04022
Random Forest	number of features: 1; number of trees: 200	0.91332	0.03965
k-NN	*k*: 2; Manhattan distance	0.90639	0.04022
K*	global blend: 3	0.89025	0.04450
MLP	learning rate: 0.1; momentum: 0.4; number of training epochs: 1000	0.88344	0.04615
Linear Regression		0.64395	0.07277

Table 2. The best results obtained with the original LMNNR algorithm

Number of prototypes	**Number of regression neighbors**	**Number of optimization neighbors**	***r***	**RMSE**
1	3	3	0.93151	0.036118
1	5	5	0.93052	0.036276
1	10	10	0.92426	0.03825
2	3	3	0.94097	0.033554
2	5	5	0.9365	0.035707
2	10	10	0.93416	0.037173
3	3	3	0.94067	0.033251
3	5	5	0.93698	0.036821
3	10	10	0.93428	0.036797
5	3	3	0.94185	0.033915
5	5	5	0.94425	0.034913
5	10	10	0.93614	0.036856

Five out of the six algorithms tested show a very good correlation of the data (~0.9) and come in a very short range from one another. Linear regression, which is included only for comparative reasons, achieves a low total correlation. This emphasizes the nonlinearity of the problem at hand. ε-SVR and Random Forest yield

the best, almost identical, predictions. kNN and K-star present similar results, despite the different metrics they use in their similarity functions.

From Table 2, it can be seen that the LMNNR results are clearly better than the results obtained by other well-established regression algorithms.

Unlike the problems studied in previous works [22, 23], it can be seen that more prototypes are needed for this particular problem. 5 prototypes provide the best results in terms of correlation coefficient. This shows that this dataset is more difficult to learn using a unique distance metric and that different regions of its input space have different characteristics with can be properly addressed with the use of prototypes.

Fig. 2 shows a comparison between the predictions of the model and the desired data, for the case with 5 prototypes, 5 regression neighbors and 5 optimization neighbors from Table 2, which yields the highest correlation coefficient *r*. One can see that the two datasets are quite close. An exception is e.g. the data point with the value of 1. Since Fig. 2 presents the results for the 10 testing sets of the cross-validation process put together, the data point with a maximum value in the test set cannot be correctly approximated by the model relying on the rest of the data in the training set. The LMNNR algorithm is based on the nearest neighbor paradigm, and therefore it cannot extrapolate to a value that is larger than any value in the training set. Furthermore, one can see that most of the data has small output values, and only 0.8% of the normalized data has output values above 0.5. This contributes to the difficulty of the model to approximate higher output values.

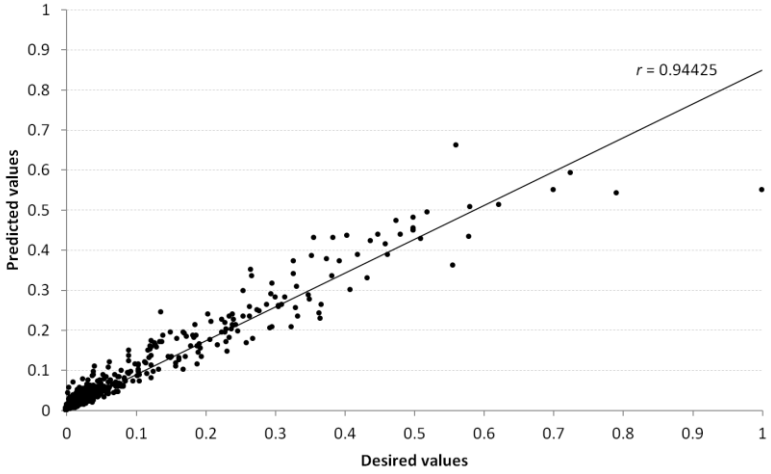

Fig. 2. Comparison between the predictions of the model and the desired data

6 Conclusions

The results obtained by the LMNNR algorithm proposed by the authors are better than those provided by other classical regression algorithms. These predictions are important for the chosen process, avoiding or, at least, minimizing the number of experiments made in toxicity conditions, and saving materials and energy. In addition,

the developed modeling methodologies can be easily adapted and applied to other chemical engineering processes.

The promising results of LMNNR determine the planning of other applications and methodologies that include this algorithm. As a future direction of investigation, one can consider its further refinement in order to automatically detect the optimal values of its parameters, namely the number of prototypes, the number of regression neighbors and the number of optimization neighbors.

Acknowledgments

This work was supported by the "Program 4, Fundamental and Border Research, Exploratory Research Projects" financed by UEFISCDI, project no. 51/2017.

References

1. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge, MA, USA (2016).
2. Witten, I. H., Frank, E., Hall, M. A., Pal, C. J.: Data Mining. 4th edn. Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Cambridge, MA, USA (2016).
3. Kaneko, H., Funatsu, K.: Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants. Chemometrics and Intelligent Laboratory Systems 137, 57-66, (2014).
4. Ge, Z., Song, Z., Wang, P.: Probabilistic combination of local independent component regression model for multimode quality prediction in chemical processes. Chemical Engineering Research and Design 92 (3), 509-521 (2014).
5. Yuan, X., Ge, Z., Song, Z.: Locally weighted kernel principal component regression model for soft sensing of nonlinear time-variant processes. Industrial & Engineering Chemistry Research 53(35), 13736-13749 (2014).
6. Ghaedi, M., Rahimi, M. R., Ghaedi, A. M., Tyagi, I., Agarwal, S., Gupta, V. K. : Application of least squares support vector regression and linear multiple regression for modeling removal of methyl orange onto tin oxide nanoparticles loaded on activated carbon and activated carbon prepared from *Pistacia atlantica* wood. Journal of Colloid and Interface Science 461, 425-434 (2016).
7. Were, K., Bui, D. T., Dick, Ø. B., Singh, B. R.: A comparative assessment of support vector regression, artificial neural networks, and random forests for predicting and mapping soil organic carbon stocks across an Afromontane landscape. Ecological Indicators 52, 394-403 (2015).
8. Baughman, D. R., Liu, Y. A.: Neural Networks in Bioprocessing and Chemical Engineering. Academic Press (1992).
9. McDonagh, J. L., Palmer, D. S., van Mourik, T., Mitchell, J. B. O.: Are the Sublimation Thermodynamics of Organic Molecules Predictable? Journal of Chemical Information and Modeling 56(11), 2162–2179 (2016).
10. Tabernero, A., Martín del Valle, E. M., Galán, M. A.: On the use of semiempirical models of (solid + supercritical fluid) systems to determine solid sublimation properties. The Journal of Chemical Thermodynamics 43(5), 711-718 (2011).
11. Leon, F., Curteanu, S.: Performance comparison of different regression methods for a polymerization process with adaptive sampling. International Journal of Computer, Electrical, Automation, Control and Information Engineering 10(10), 1515-1519 (2016).

12. Pirdashti, M., Movagharnejad, K., Mobalegholeslam, P., Curteanu, S., Leon, F.: Phase equilibrium and physical properties of aqueous mixtures of poly (vinyl pyrrolidone) with trisodium citrate, obtained experimentally and by simulation. Journal of Molecular Liquids 223, 903-920 (2016).
13. Hlihor, R. M., Diaconu, M., Leon, F., Curteanu, S., Tavares, T., Gavrilescu, M.: Experimental analysis and mathematical prediction of Cd(II) removal by biosorption using support vector machines and genetic algorithms. New Biotechnology 32(3), 358-368 (2015).
14. Chelariu, R., Suditu, G. D., Mareci, D., Bolat, G., Cimpoesu, N., Leon, F., Curteanu, S.: Prediction of corrosion resistance of some dental metallic materials with an adaptive regression model. The Journal of The Minerals, Metals & Materials Society (JOM) 67(4), 767-774 (2015).
15. Curteanu, S., Smarandoiu, M., Horoba, D., Leon, F.: Naphthalene sublimation. Experiment and optimisation based on a neuro-evolutionary methodology. Journal of Industrial and Engineering Chemistry 20(4), 1608-1611 (2014).
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H.: The WEKA data mining software: an update. ACM SIGKDD Explorations 11(1), 10–18 (2009).
17. Rumelhart, D. E., Hinton, G.E., Williams, R. J.: Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1, MIT Press Cambridge, MA, USA (1986).
18. Smola, A. J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing 14(3), 199-222 (2004).
19. Cleary, J. G., Trigg, L. E.: An instance-based learner using an entropic distance measure. In: 12th International Conference on Machine Learning, pp. 108–114. (1995).
20. Painuli, S., Elangovan, M., Sugumaran, V.: Tool condition monitoring using K-star algorithm. Expert Systems with Applications 41(6), 2638–2643 (2014).
21. Breiman, L.: Random Forests. Machine Learning 45(1), 5-32 (2001).
22. Leon, F., Curteanu, S.: Evolutionary algorithm for large margin nearest neighbour regression. In: 7th International Conference on Computational Collective Intelligence Technologies and Applications, pp. 286-296. (2015).
23. Leon, F., Curteanu, S.: Large margin nearest neighbour regression using different optimization techniques. Journal of Intelligent & Fuzzy Systems 32, 1321-1332 (2017).
24. Weinberger, K. Q., Saul, L. K.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, 207–244 (2009).