Enhancement of Classification of Small Data Sets Using Self-awareness - An Iris Flower Case-Study

Hedyeh A. Kholerdi, Nima TaheriNejad, and Axel Jantsch
Institute of Computer Technology
TU Wien, Vienna, Austria
Email: {hedyeh.kholerdi, nima.taherinejad, axel.jantsch}@tuwien.ac.at

***Abstract*—In big-data (Deep) Neural Network (NN) algorithm is often used for classification. However, such a massive mine of data is not always available and a shortage of training data can significantly deteriorate the performance of NNs and other classifiers. Therefore, we propose a self-aware multiple classifier system suitable for "Small-Data" cases. This algorithm uses self-awareness to switch between classifiers to improve its performance. We tested the algorithm for the classification of iris flower species using the Iris standard database. Compared to NN, our algorithm showed up to 17% classification success rate improvement with up to 10 times smaller standard deviation.**

I. Introduction

Lack of sufficient and efficient data set prevents machine learning applications such as classification and prediction from being generalized and accurate [1]. They behave accurately during training, however, the test error can increase due to over-fitting [2]. Therefore, there has been growing attention to the development of the learning techniques to enhance the generalization and accuracy of these applications [1], [3], [4]. Onisko et al. [3] used Bayesian network multiple-disorder model to diagnose liver disorders. A drawback there is the assumption of probability distribution and consequently setting of respective parameters. Since they could not be learned from the small data set, the performance heavily depends on the probability of the assumed distribution being close to the real distribution of the data. Shin et al. [5] demonstrated that Support Vector Machine (SVM) outperforms Back-Propagation Neural Network (BPN) on bankruptcy prediction problem when the size of data set is small. Therefore, despite their dominance, NNs do not always have the best performance and the use of an appropriate learning algorithm improves the performance, especially in the case of small data.

Chopra et al. dealt with the issue of small data for a face recognition application in which their solution is to learn a similarity metric from data [6]. They used a discriminative loss function on an energy-based model and attempted to minimize it in order to be able to use the similarity metric later to match new faces. A study in [2] suggests the unsupervised learning in deep learning applications when the size of labeled data set is small. Unsupervised pre-training techniques have shown to improve the generalization. The authors believe that certain feed-forward neural network can serve the same purpose [2].

In continuation of the previous attempts to improve both the generalization and accuracy, the idea of a multi-classifier technique for the small data set which uses self-awareness to switch between algorithms is proposed in this work. Due to different mechanisms of learning, some information of the data is often overlooked in each individual learning algorithm. Therefore, using multiple algorithms has the advantage of combining different information (correlations) extracted from the data by each algorithm. As a result, a multiple classifier system makes the best use of the data, even in a small batch.

II. Literature Overview

Multiple Classifier Systems (MCSs), mostly known as classifier ensembles, have been studied over two decades [7].They combine a set of individual classifiers in order to improve the performance and reliability of the classification in a variety of applications such as digital signal processing and pattern recognition [8], [9]. Every classifier such as NN, SVM and decision tree achieves an accurate result in specific fields and under some constraints. Therefore, MCS inherits the desired features of every single classifier and is able to deal with highly complex problems, uncertainty, high-dimensionality of data, optimization problems, and so on [10].

The goal of an MCS is to create a superior classifier which outperforms its components. The key to reaching higher performance or accuracy appears in the diversity of classifiers [9], [11]. Therefore, the selection of appropriate classifiers is considered to be the first concern. The second concern addresses the strategy of integration of chosen classifiers. Some focus on the input fusion, some combine the outputs and there are some which have both input and output fusion.

Three major categories of MCS approaches include training sample manipulation, parallel combination, and concatenation combination [9]. Bagging and Boosting algorithms are popular MCS methods based on the manipulation of training samples which create multiple training data sets to generate multiple hypotheses [8]. Each data set is used to train the learning algorithm. On the other hand, parallel combination independently trains the chosen classifiers and then uses some strategies such as majority voting to combine the outputs [12]. In contrast, concatenation combination consists of a chain of classifiers in which the output of one is fed to the next one as an input [13].

Some classifiers can provide the ranking score for the classes of a given problem or a value of probability or confidence. In concatenation architecture, the ranking from one classifier is used to refine the number of classes and therefore

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

the most confident classes are sent to the next classifiers [14]. Another example of ranking score uses a model such as regression model from the vector of ranking scores and the outputs of the classifiers enter the model for the final decision [7]. There has not been noticeable attention to ranking approaches probably due to the inability of most classifiers in producing the ranking score. Moreover, this approach is mostly preferred in the domain of pattern recognition [15].

Multiple classifier systems usually sacrifice fast computation for higher accuracy by running multiple classifiers. Therefore, they result in high computational costs. We propose an approach which possesses the benefits of MCS in high performance and accuracy as well as low computational cost. This approach aims specifically at dealing with those classification applications which suffer from the small size of data sets. Therefore, the proposed algorithm, by exploiting several classifiers, improves the accuracy and generalization of the classification for the small data set. All the while it reduces the computational cost of MCS, by keeping the number of classifiers it runs to a minimum and avoiding to run additional classifiers when not necessary.

III. Self-aware Multiple Classifier System (MCS)

Self-awareness is the ability of the system to monitor its state, behavior, and performance to update one or more of its components to achieve its goals [16]. However, monitoring has received little attention so far. Recently, TaheriNejad et al. [17] published a study on various elements of observation and their potential role in self-awareness. This led to further research which showed the benefits of these elements, such as data reliability [18], [19] and attention [19], [20], in different applications. One of these elements of observation is 'confidence' which caters the system with a measure of reliability of the results of an algorithm. This helps the self-aware system to take better decisions based on the reliability of its subsystems, and use its resources in a more efficient manner and based on the situation at hand. We have taken advantage of this concept in the proposed algorithm and show how it improves the performance and the reliability of the system, especially in the case of small-data, while keeping the number of classifiers it runs to a minimum.

Let us assume that $T_{x_i}^{k_l}$ is the True Class of a sample x_i (Ground Truth) for the specific class of k_l and $E_{x_i}^{A_j}$ is the class estimated by algorithm A_j for x_i. Then, we define the confidence of A_j for this classification equal to the probability of $E_{x_i}^{A_j}$ being equal to $T_{x_i}^{k_l}$. That is,

$$c\Big(A_j(x_i, k_l)\Big) = p\Big(E_{x_i}^{A_j} == T_{x_i}^{kl}\Big), \tag{1}$$

where $p\big(E_{x_i}^{A_j} == T_{x_i}^{kl}\big)$ is provided by each algorithm. In other words, the confidence of the algorithm shows the likelihood of a correct classification. Therefore, if this value is below a certain threshold, c_{th}, it is wise to check for alternative methods of classification. Otherwise, it is better to avoid running other classifiers in order to avoid the extra computation load of other algorithms which make MCSs heavy and resource hungry. Hence comes the proposed algorithm shown in Fig. 1. In the proposed approach any classification algorithm can be used. In our experiments, described in section IV, we use NN, SVM and Naive Bayesian (NB) classifiers. The ranking (best, second best, last algorithm) of different algorithms is determined in the cross-validation phase.

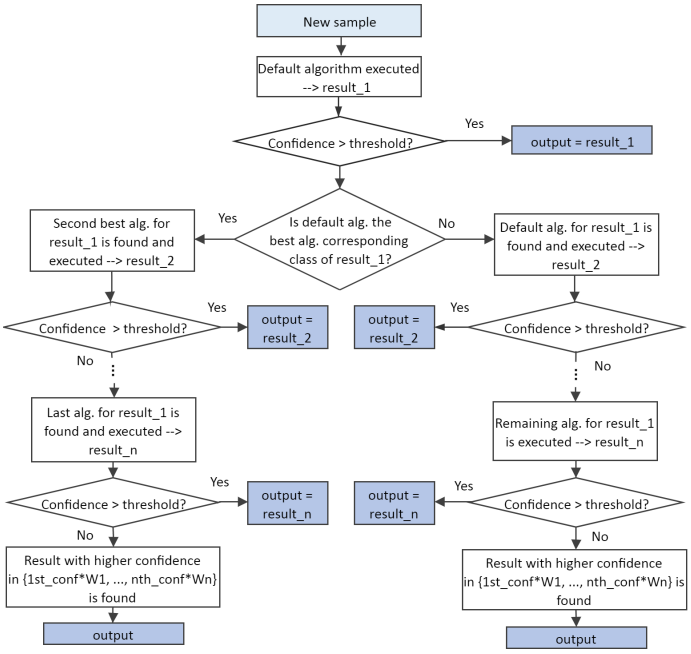


Fig. 1. The flowchart of proposed algorithm for self-aware classification.

As we can see in Fig. 1, every time a used classifier is not confident enough, other algorithms are used for classification based on their rank for the estimated class. This ranking is based on the overall confidence of all algorithms for each class k_l, which is calculated by

$$c\Big(A_j^{k_l}\Big) = \frac{1}{n}\sum_{i=0}^{n} c\big(A_j(x_i, k_l)\big), \tag{2}$$

where n is the number of all samples classified during the cross-validation phase as belonging to the class k_l. As shown in Fig. 2, during the cross-validation phase the ranking in each class, $c(A_j^{k_l})$, is calculated for all algorithms and then they are sorted based on their confidence for each class.

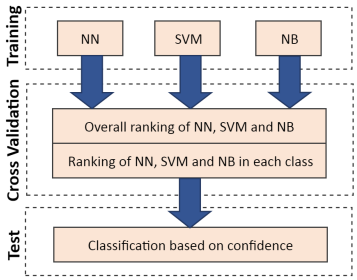


Fig. 2. The overall architecture of the proposed classification system.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

To select the default algorithm, during the cross-validation the overall confidence of each algorithm is calculated over all m possible classes;

$$C_{A_j} = \frac{1}{m} \sum_{k_l=0}^{m} c\big(A_j^{k_l}\big). \quad (3)$$

The default algorithm, A_d, is the one with the largest overall confidence. In other words, $C_{A_d} = \max\{C_{A_j}|_{j=0}^{l}\}$, where l is the total number of algorithms, in our case three ($l = 2$).

In summary, the proposed self-aware classifier starts with the default algorithm, A_d, which classifies the sample x_i as belonging to the class k_l with the confidence of $c(A_d^{k_l})$. If $c(A_d^{k_l}) < c_{th}$ (in the experiments we use $c_{th} = 0.9$), the best classifier for the class k_l is selected, as identified in the cross-validation phase (if A_d is already the best algorithm for the class k_l the second best algorithm, A_s, will be selected). If it does not provide sufficient confidence, the remaining algorithms (A_r) are used one by one until the confidence is above the threshold. If none of the algorithms provides a confidence above the threshold, the estimated class with the highest confidence, c_{max}, is selected. However, for this selection we consider the overall reliability of various algorithms (overall confidence, C_{A_j}), and therefore, we have heuristically tipped the scale by giving the best algorithm 5% of advantage and the worst algorithm 5% of disadvantage. That is, the maximum confidence is calculated using

$$c_{max} = \max\{1.05\ c(A_d^{k_l}),\ c(A_s^{k_l}), 0.95\ c(A_r^{k_l})\} \quad (4)$$

and the class identified by the algorithm with maximum confidence (c_{max}) is considered as the result.

IV. Simulation and Results

A. Simulation Setup

In our experiments we use three classifying algorithms, namely NN, SVM and NB. NN is a network of neurons which includes the input and output layers as well as a number of hidden layers whose connections are weighted [21]. The goal of training the network is to find the values of weights and generally improve the internal structure of the network. During training, the network adapts to minimize the error on the training samples. The basic idea of SVM as a binary classifier is to map the input vectors into high-dimensional feature vectors in order to optimize the separating hyperplanes [22]. SVM uses linear or nonlinear models to generate the decision function in the form of hyperplanes. The hyperplanes have the maximum margin from the classes they are separating. Using the training set, the most optimal hyperplane is produced. Finally, Bayesian classifier is a statistical classifier, based of Bayes' theorem, which estimates the probability that a sample belongs to each class [23]. The NB classifier considers that the values of the attributes are conditionally independent of a given class.

The proposed MCS with these three classifying algorithms has been tested on an Iris data set from the UCI Machine Learning Repository [24]. The data set includes 150 samples with four attributes of Sepal and Petal, width and length. Each sample belongs to one of three species of Iris flower. The implementation has been done using MATLAB®. The toolboxes of NN, SVM and NB have been used and these algorithms, as binary classifiers, have been customized in order for the maximum performance to be achieved. The NN has 15 hidden layers and uses the scaled conjugate gradient method over mean squared normalized error performance function to update the weights and bias values. The SVM was designed as a multi-class Support Vector Machine with standardized predictor matrix and one learner. Automatic hyperparameter optimization was used to minimize the five-fold cross-validation loss. Since the toolbox provides the acquisition function option, it was set to 'expected-improvement-plus' for reproducibility. Finally, NB was trained as a multi-class naive Bayes model with default normal kernel distribution for predictors.

TABLE I
A sample set of results for the smallest run (20) and the largest run (70). All runs include 10 iterations.

Run	20 (20, 13, 40)			70 (70, 40, 40)		
Alg.	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
SVM	88.50	91.25	10.08	93.50	92.50	4.74
NB	93.25	92.50	4.09	**95.75**	**96.25**	4.09
NN	77.00	85.00	20.10	95.00	95.00	4.08
Prop.	**94.25**	**93.75**	**2.65**	95.25	**96.25**	**3.62**

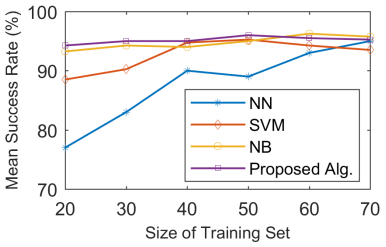
The simulation results have been calculated over 10 iterations of the proposed algorithm for each run (setting of training data size). At the beginning of each iteration, the data set is randomly disordered. Six settings of training sizes, namely $\{70, 60, 50, 40, 30, 20\}$ have been tested. The ratio of the training size to the size of cross validation set was considered around 1.7 which leads to $\{40, 35, 30, 25, 17, 13\}$ for the respective runs. The size of the test set, however, was fixed to 40 samples for all settings. Another parameter of the simulation is the threshold of confidence which was experimentally set to 90% for all classes.

B. Results and Comparison

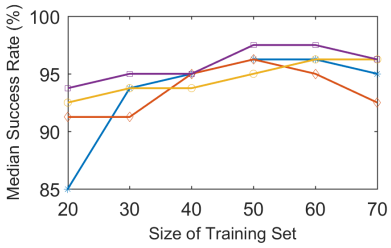
For the sake of brevity and clarity, in Table I we have inserted the detailed numerical results only for the 20 and 70 runs. Instead, we chose to present the results in Fig. 3 to Fig. 5, where the trends and changes can be better and easier seen, analyzed, and understood.

The proposed system, runs on average 1.27 algorithms for each classification. As we can see in Fig. 3(a), the flat curve of the proposed algorithm shows that it maintains a relatively constant success rate during all runs, regardless of the size of the training data. This is backed-up with the standard deviation which is depicted in Fig. 4. Hence, we can conclude that the proposed algorithm is highly reliable, and regardless of the size of the training data, its results are most likely very close to the best performance possible. We can observe this in the clean and concentrated histogram of the result distribution of the proposed algorithm, at the bottom of Fig. 5.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE



(a) Mean



(b) Median

Fig. 3. The graphs of the mean and median values of the success rate for different runs, each calculated over 10 iterations.

In comparison with other algorithms, we observe that in the majority of runs, the proposed algorithm outperforms other algorithms. This superiority is absolute and larger in the case of smaller data (in this case for the 50 run and smaller runs). For example, as we see in Table I, the mean of the proposed algorithm at the 20 run is 94.25 which 17% higher than that of the NN (77.00) while having a 7.6 times smaller standard deviation (2.65 compared to 20.10). Although the gap between mean success rate decreases with the increase of the training size (Fig. 3(a)), at the 40 run the standard deviation between the two, as shown in Fig. 4, increases to a 10.5 times difference (21.43 for the NN to 2.04 for the proposed algorithm). The best algorithm among the single classifiers is the NB which closely follows the proposed algorithm and even surpasses it by a margin 0.75% and 0.5% in the mean success rate of the 60 and 70 runs. That is, as we see in Table I, for 70 runs the average of NB is 95.75 compared to that of the proposed algorithm whereas they both have a median of 96.25. In terms of standard deviation, NB is only better in the case of the 60 run by having 0.78 of the standard deviation of the proposed algorithm. In the case of median[1], on the other hand, as depicted in Fig. 3(b), the proposed algorithm has the best success rate in all cases.

In summary, our experiments show that the proposed algorithm is superior for small-data classification. That is, when there is only a small set of data available, the proposed algorithm can improve the overall performance (mean success rate) and be more reliable (showing a smaller standard deviation). Although this gap, especially in terms of mean success rate, closes with the increase in the size of the training data, the proposed algorithm is still a more reliable algorithm (smaller standard deviation). Therefore, its consistent performance makes it a suitable candidate across the range. Last but not least, in all cases, it is most likely that the proposed algorithm

[1]Median is the middle value in an ordered list of data [25]. In our case, that is the assorted success rates of each algorithm.

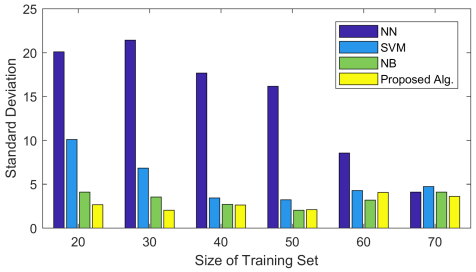


Fig. 4. The standard deviation of the success rate over 10 iterations.

provides the best (a near-optimal) answer consistently and at each try thanks to having a higher median throughout (which is closer to the average) and a smaller standard deviation. This can be seen clearly in Fig. 5 where the histogram shows the distribution of the success rate of all four algorithms in each iteration of the 20 runs. We observe that the results of the proposed algorithm are concentrated in the upper tenth of the chart, while others have a wider distribution (in the case of the NN even in the 4th tenth). This is particularly significant, because in most practical applications the classification is done only through one iteration, hence, the quality of the result at each iteration is very crucial and having an algorithm which most likely produces the best or a near-optimal answer is of paramount importance.

V. Conclusion

We have proposed a MCS that selects one of the available classifiers based on a confidence metric. The confidence, defined as the probability of correct classification, is computed for each classifying algorithm during the test session. We showed that, by using confidence as a classifier selection criterion, we can outperform each individual classifier while avoiding the cost of running several classifier algorithms (on average the proposed system run only 1.27 algorithms per sample). In a case study we used a Neural Network, Support Vector Machine, and a Naive Bayes model as component classifiers, and applied our proposed self-aware MCS algorithm on an data repository with Iris flower images and found out that our proposed algorithm performs better in almost all tested cases and is particularly accurate when small data sets are used.

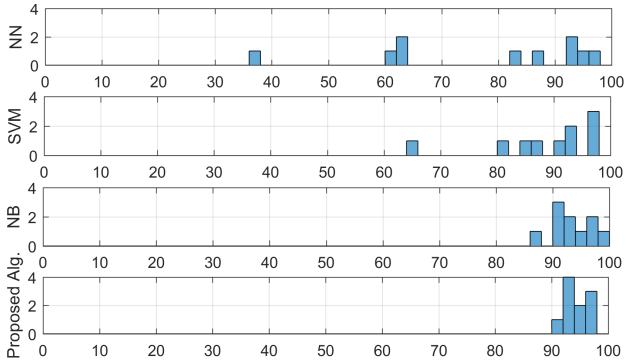


Fig. 5. The success rate histograms of all algorithms in the 20 runs, where the x axis shows the success rate (at intervals of 2%). Each column shows the number of iterations (out of 10) which had the respective success rate.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Agnieszka Oniśko, Marek J Druzdzel, and Hanna Wasyluk. Learning bayesian network parameters from small data sets: Application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

[4] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[5] Kyung-Shik Shin, Taik Soo Lee, and Hyun-jung Kim. An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, 28(1):127–135, 2005.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[7] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE transactions on pattern analysis and machine intelligence*, 16(1):66–75, 1994.

[8] Thomas G Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[9] Peijun Du, Junshi Xia, Wei Zhang, Kun Tan, Yi Liu, and Sicong Liu. Multiple classifier system for remote sensing image classification: A review. *Sensors*, 12(4):4764–4792, 2012.

[10] Michał Woźniak, Manuel Graña, and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17, 2014.

[11] Robert PW Duin and David MJ Tax. Experiments with classifier combining rules. In *International Workshop on Multiple Classifier Systems*, pages 16–29. Springer, 2000.

[12] Jinxiu Qu, Zhousuo Zhang, and Teng Gong. A novel intelligent method for mechanical fault diagnosis based on dual-tree complex wavelet packet transform and multiple classifier fusion. *Neurocomputing*, 171:837–853, 2016.

[13] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, Francesco Tortorella, and Mario Vento. A cascaded multiple expert system for verification. In *International Workshop on Multiple Classifier Systems*, pages 330–339. Springer, 2000.

[14] Didier Guillevic and Ching Y Suen. Hmm-knn word recognition engine for bank cheque processing. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1526–1529. IEEE, 1998.

[15] Ching Y Suen and Louisa Lam. Multiple classifier combination methodologies for different output levels. In *International workshop on multiple classifier systems*, pages 52–66. Springer, 2000.

[16] Nikil Dutt, Axel Jantsch, and Santanu Sarma. Toward smart embedded systems: A self-aware system-on-chip (soc) perspective. *ACM Trans. Embed. Comput. Syst.*, 15(2):22:1–22:27, February 2016.

[17] Nima TaheriNejad, Axel Jantsch, and David Pollreisz. Comprehensive observation and its role in self-awareness - an emotion recognition system example. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, Gdansk, Poland, 2016.

[18] Maximilian Götzinger, Nima Taherinejad, Amir M. Rahmani, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen. *Enhancing the Early Warning Score System Using Data Confidence*, pages 91–99. Springer International Publishing, Cham, 2017.

[19] Arman Anzanpour, Iman Azimi, Maximilian Götzinger, Amir M. Rahmani, Nima TaheriNejad, Pasi Liljeberg, Axel Jantsch, and Nikil Dutt. Self-awareness in remote health monitoring systems using wearable electronics. In *Proceedings of Design and Test Europe Conference (DATE)*, Lausanne, Switzerland, March 2017.

[20] N. TaheriNejad, M. A. Shami, and S. M. P. D. Self-aware sensing and attention-based data collection in multi-processor system-on-chips. In *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 81–84, June 2017.

[21] Jacek M Zurada. *Introduction to artificial neural systems*, volume 8. West St. Paul, 1992.

[22] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[23] K Ming Leung. Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*, 2007.

[24] PM Murphy and DW Aha. Uci repository of machine learning databases, university of california, department of information and computer science, irvine, ca, 1994.

[25] Michael Havbro Faber. *Statistics and probability theory: in pursuit of engineering decision support*, volume 18. Springer Science & Business Media, 2012.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE