TSSweb: a Web Tool for Training Set Selection

Giovanni Acampora and Autilia Vitiello
Department of Physics "Ettore Pancini"
University of Naples Federico II
Naples, Italy
{giovanni.acampora, autilia.vitiello}@unina.it

***Abstract*—Supervised learning methods aimed at performing precise predictions by learning from labeled training data. Unfortunately, training data can contain noisy or wrong information, specially when they come from real-world applications. In this scenario, applying a so-called training set selection procedure on data can lead to improve the performance of the supervised learning methods used for classification or regression tasks. In literature, several training set selection techniques have been proposed, but, to the best of our knowledge, few software tools implement this procedure. Moreover, all of them require programming capabilities or software package installation what makes their use difficult for people without specific computer skills. This paper proposes the first web-based tool, named TSSweb, for performing an accurate selection of the training instances. Thanks to its web nature, TSSweb enables all researchers, coming from several and heterogeneous scientific backgrounds, to reduce own datasets so as to improve their analysis and reduce the execution time of their supervised learning models. As shown in the experimental session, TSSweb produces reduced datasets with a good quality as well as being user-friendly.**

I. Introduction

In the machine learning area, supervised learning methods aimed at building mathematical models based on sample data, known as *training data*, in order to perform predictions within classification or regression problems. Unfortunately, in many real-world applications, training data can contain noisy or wrong information and also the performance of best supervised learning methods could worsen when they deal with these data [1]. Training Set Selection (TSS) [2] represents a suitable and consolidated approach to face these problems. Precisely, TSS consists of a preprocessing technique that selects only the relevant dataset instances before performing training for classification or regression tasks [3]. Thanks to the computation of a reduced training dataset including the most adequate instances, TSS techniques achieve a twofold benefit: on one hand, the performance of the supervised learning methods can be improved, while, on the other hand, the execution time can be decreased.

In literature, there are several approaches for addressing the TSS problem as described in [4]. However, the most part of them do not provide software to enable researchers to run the proposed procedure. Among the software, it is possible to count packages in Java[1] and R[2] languages such as those provided in [5] [3]. However, these approaches are usable only by people with programming skills and, typically, researchers far from computer science area do not have these capabilities. To address this issue, one of the most known tools for machine learning, namely KEEL [6] [7] [8], provides a graphical interface to make machine learning tasks such as the training set selection more simple. However, this tool works after performing some installation activities including the Java Virtual Machine set up and, sometimes, people are not practical and confident with these installation procedures.

Starting from these considerations, this paper proposes the first web-based tool, namely TSSweb, aimed at performing the TSS procedure in an easily way regardless from the computer skills of users. In detail, the implemented procedure is a so-called *wrapper* training set selection mechanism, i.e., it requires the use of an embedded supervised method to detect the most adequate instances to compose the reduced training dataset. In other words, the quality of the selected instances is determined by the performance measure that a supervised approach obtains when these instances are used to train it. In this work, the best performance value is searched through the application of an evolutionary algorithm, namely a genetic algorithm. The proposed TSS method handles both classification and regression problems. Hence, TSSweb allows users to run the TSS method by using both classifiers and regressor methods. In particular, the classifiers included in TSSweb are: the Linear Discriminant Analysis (LDA), the K-nearest neighbour (KNN), the Support Vector Machine (SVM), the Multi-Layer Perceptron (MLP) and the Decision Tree (DT). As for regressors, TSSweb includes the Linear Regressor (LR) and the regression version of the K-nearest neighbour (KNN-R), Support Vector Machine (SVM-R), the Multi-Layer Perceptron (MLP-R) and the Decision Tree (DT-R).

TSSweb is accessible to everyone regardless from their computer skills thanks to a user-friendly web interface. At the same time, it produces good reduced datasets as shown in the wide experimental session where two quality indicators are used for the evaluation. In detail, the first one determines the goodness of the reduced dataset in representing the original dataset, whereas, the second one studies the efficacy of the proposed TSS method with respect to two baseline approaches.

The remaining of the paper is organized as follows. Section II describes the TSS problem and its formulation as an optimization problem to be maximized. Section III is the core of the paper where the proposed tool is described in a detailed way. Before concluding in the Section V, the experimental session and its results are reported in the Section IV.

[1]https://www.java.com/it/
[2]https://www.r-project.org/

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

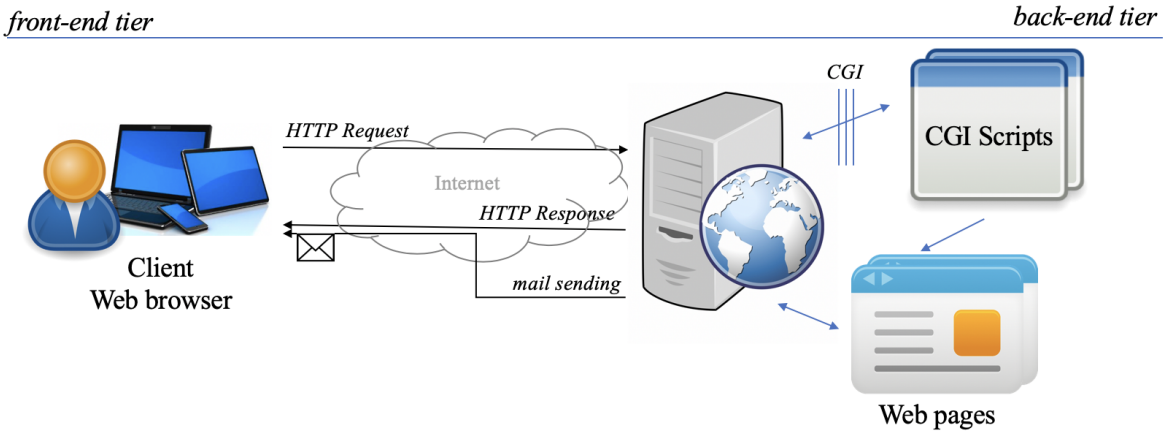


Fig. 1: TSSweb architecture

II. Training set selection problem

The Training Set Selection (TSS) is a pre-processing method used to select relevant instances in training data before applying a supervised method. TSS is very useful in many real-world applications, where datasets can contain noisy or wrong information that make difficult the classification or regression also for the best methods. Formally, let TR be the original training set composed of n instances. Each instance I_i is a pair (x_i, y_i) with $i = 1, \ldots, n$, where x_i defines an input vector of attributes and y_i defines the corresponding class label [3]. Each input vector contains m input attributes that are quantitative or qualitative information that describe the corresponding instance. The goal of any TSS technique is to produce a set of instances $S \subseteq TR$ to be used to train a supervised learning method capable of predicting new instances with the same or higher quality of the same learning method trained with the original training data TR. Hence, it is possible to define the TSS problem as an optimization problem. Let us consider TR be the original training set and Λ_{TR} be the set of all possible subsets S of TR, the TSS problem can be formulated as below:

$$\max \mathbf{y} = f(S) \text{ with } S \in \Lambda_{TR} \tag{1}$$

where $f : \Lambda_{TR} \rightarrow \mathbb{R}$ is defined as follows:

$$f(S) = eval(TR, S) \text{ with } S \in \Lambda_{TR} \tag{2}$$

where $eval$ is a function that computes the quality (in percentage) of the prediction performed by a supervised learning method by considering TR as testing set and S as the training set. The cardinality of the set S should be smaller than the cardinality of the set TR in order to reduce the supervised learning method time complexity. The amount of reduction of S with respect to TR is referred to as *reduction rate*. Formally,

$$red(TR, S) = \frac{|TR| - |S|}{|TR|} \cdot 100 \text{ with } S \in \Lambda_{TR} \tag{3}$$

where red is a function that computes the reduction rate in percentage of the set S with respect to the original set TR.

In literature, there are several training set selection methods and they are categorized in filter and wrapper approaches [9]. The filter methods use general criteria to select the instances regardless of the supervised method that will be applied. Instead, the wrapper methods use a complete computation of the supervised method in order to select the most suitable instances. Hence, typically, filter methods are faster but less accurate with respect to wrapper methods. Since the computation time of the training is not an issue for TSSweb which will be run on a powerful server and send results also in an asynchronous via by email, TSSweb implements a wrapper training set selection mechanism. More details are given in the next section.

III. TSSweb

TSSweb is a web-based tool that permits 1) to perform a training set selection procedure on a given dataset by using a set of user's inputs, 2) to obtain the corresponding reduced training dataset; 3) to display the quality of the selected supervised method trained with the reduced dataset, if a test dataset is given in input; 4) to compare the obtained quality with two baseline approaches. As aforementioned, TSSweb implements a wrapper training set selection procedure, i.e., the selection of the most suitable instances is based on the computation of an embedded supervised learning method and the quality that it achieves. In this work, the proposed training set selection exploits the evolution of a genetic algorithm to search the best subset of instances. The computational effort of this procedure is given by the number of solutions evaluated. The TSS procedure addresses both regression and classification problems. In other words, the dataset uploaded to be reduced can contain both a numerical and categorical target variable. TSSweb is accessible by means of any web browser and makes available a user-friendly interface where it is possible to fill a form with all necessary inputs. Once the user submits the form, TSSweb manages the request of the user thanks to the architecture displayed in Fig. 1. In detail, the web browser contacts the web server using the HTTP connection over Internet. In turn, the web server receives the

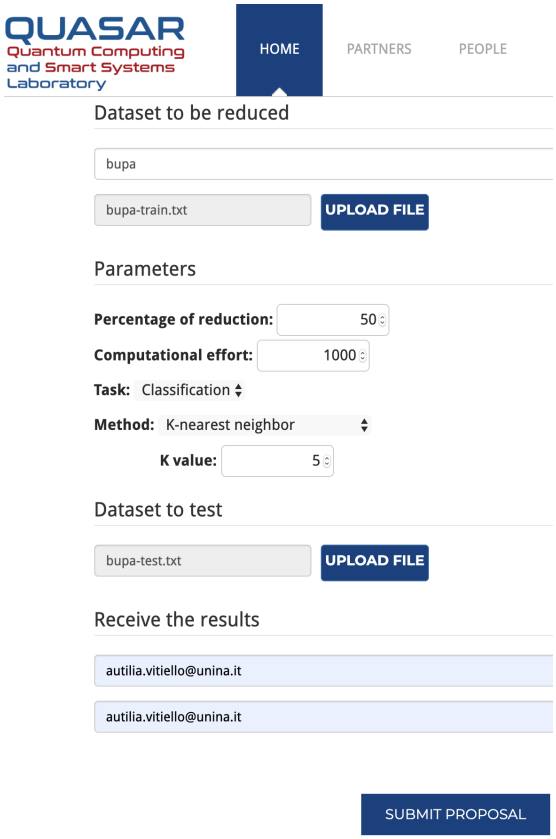


Fig. 2: TSSweb interface

request and calls a Common Gateway Interface (CGI) script. The CGI script performs the training set selection procedure on the given dataset by using the other user's input information. Then, it builds a HTML page with the dynamically obtained results that will be displayed to the user. Moreover, as well as memory resource, the CGI script uses other hardware resources of the web server such as the email server SMTP in order to send results to user's email address, too. Hereafter, a description of the the two tiers of the TSSweb architecture is given.

A. *Front-end tier*

TSSweb is accessible by means of any web browser at the following link: http://quasar.unina.it/trainingSetSelection.html. The graphical web interface, shown in Fig. 2, allows the users to introduce all inputs necessary to run the training set selection procedure implemented on server-side. Precisely, the inputs of the form include:

- the path of the file containing the original dataset to be reduced. Thanks to this information, the file is uploaded and processed on server-side;
- the percentage of reduction thanks to which the number of instances that will compose the reduced dataset is computed. The value of this percentage ranges from 1% to 99%;
- the computational effort representing the effort used to perform the task of the TSS procedure. This value ranges from 10^2 to 10^6 and it corresponds to the number of solutions evaluated before giving in output the most adequate reduced dataset. The higher the computational effort, the longer is the time that the user should wait to obtain the results. On the other side, the higher the computational effort, the results will be better. For this reason, this parameter is left to the user;
- the choice of the task between classification and regression. Indeed, TSSweb manages datasets whose the target variable is numerical (in the case of regression) or categorical (in the case of classification). This information is necessary to run different supervised learning methods according to the task;
- the supervised learning method to be used to select the most adequate instances. It is possible to select among five methods for classification and as many methods for regression;
- hyper-parameters of the selected supervised learning method. In detail, it is possible to insert the following information for each supervised learning method.
  - **K-nearest neighbour**: the K value (from 1 to 11) both in the classification and regression case;
  - **Support Vector Machine**: the C value (from 10^{-6} to 10^5), the *tolerance error* (from 10^{-7} to 0.1), the *maximum number of iterations* (from 1 to 10^4) and information about the kernel both in the classification and regression case. Among kernels, it is possible to select the *linear*, the *polynomial* or the *gaussian* one. Then, for the polynomial kernel it is possible to set the *degree* (ranging from 1 to 10), whereas, for the gaussian one it is possible to set the *gamma* value (ranging from 10^{-6} to 10^5);
  - **Multi-layer perceptron**: the *number of hidden layers* (from 1 to 5), the *number of neurons* for the hidden layers (from 1 to 100), the *tolerance error* (from 10^{-7} to 0.1), the *maximum number of iterations* (from 1 to 10^4) and the *learning rate* (from 10^{-6} to 10^4) both in the classification and regression case;
  - **Decision Tree**: the *maximum depth* (from 1 to 500) and the *criterion* to measure the quality of a split (*gini* and *entropy* in the classification case and the *Mean Squared Error* (MSE) and the *Mean Absolute Error* (MAE) in the regression case);
  - **Linear Discriminant Analysis** and **Linear regression**: no hyper-parameters are necessary.
- the path of the file containing the test dataset (optional). This file is used to test the performance of the selected supervised learning method trained by using the reduced dataset and to compare the results with two baseline approaches as described in Section IV;
- the user email address to receive results in an asynchronous way.

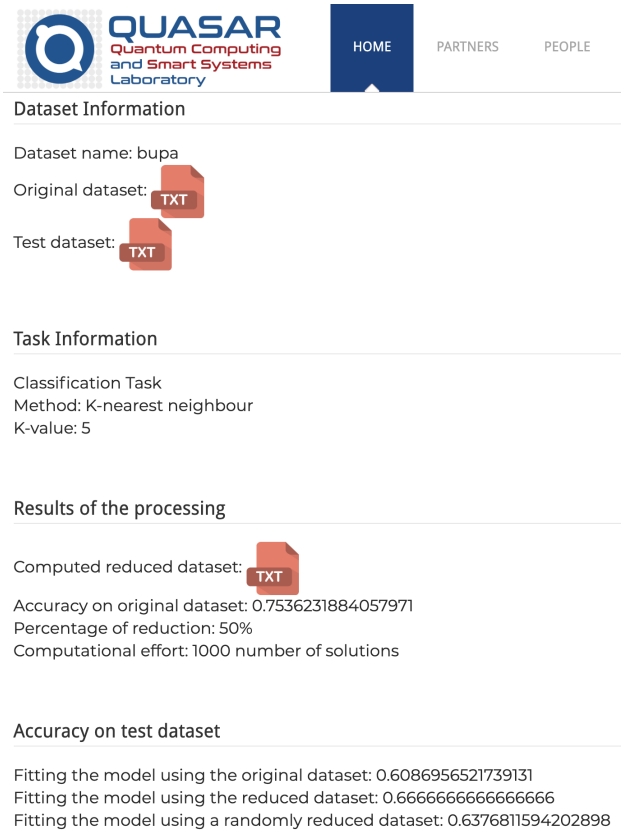


Fig. 3: TSSweb report web page

Once the request is submitted, the web server handles the request and computes a response in the form of a web page. The displayed web page contains a report of the training set selection procedure. More precisely, this web page, shown in Fig. 3, allows users to download the reduced dataset and to visualize some preliminary results related to its quality. Moreover, since the training set selection procedure could work also for several minutes, TSSweb uses the user's email address given in input to send him/her the results of the procedure. So, it is not necessary that users wait the results appear on web page. The technologies used to implement the graphical interface on front-end tier are HTML5[3], Css[4] and Javascript[5].

B. back-end tier

The web server receives the request by the web client and handles it by calling a CGI script written in Python[6]. CGI is an interface specification for web servers to execute programs like console applications running on a server that generates web pages dynamically. In our case, the CGI script computes the reduced training dataset by applying a wrapper TSS method. As described in Section II, TSS problem can be mapped in an optimization one to be maximized. Hence, evolutionary algorithms seem to be a suitable solution to address this task [10] [11]. In general, the main concept of the evolutionary wrapper TSS methods is that they maintain a population of individuals, which are subsets of instances, i.e., candidate reduced datasets. Specifically, they operate on encoded representations of the solutions, called *chromosomes*, that correspond to the representations of individual features in nature. The algorithm evolution starts from a population of randomly generated individuals and consists in successive *generations* [12]. In each generation, as in nature, a *selection process* provides the mechanism for selecting better solutions to survive. Each solution is evaluated by means a *fitness function* that reflects how good it is, compared with other solutions in the population. The better is the fitness value of an individual and higher are its chances of surviving. In the case of TSS, the fitness of an individual S is based on the value $eval(S)$ defined in Eq. 2. Among evolutionary algorithms, TSSweb runs a simple genetic algorithm [13] whose the pseudo-code is shown in the table I. Hence, the recombination of genetic material is simulated through two operators: *crossover* that exchanges portions between two randomly selected chromosomes and *mutation* that causes random alteration of the chromosome genes. The algorithm evolution ends when specified conditions are reached. In our algorithm, the evolution ends when the maximum number of evaluations of fitness (corresponding to the number of solutions evaluated) is achieved. The output is represented by the subset of instances in the final population with the best fitness value. To conclude with the description of our genetic algorithm applied to solve the TSS problem, hereafter, we describe the solution encoding for our TSS problem and the recombination and selection operators used. As described in Section II, a solution of the TSS problem should represent a subset of TR. Other papers [3] [9] describe a solution as a vector of n variables (one for each instance in the training set TR) where each variable can be set to two possible states: 0 and 1. In detail, if the variable is set to 1, the corresponding training instance is included in the subset of TR, otherwise, it is not included. However, this technique leads to have a large chromosome size. Therefore, in this paper, we decide to consider each instance of the dataset to be reduced identified by an index l (with $l = 0, 1, \ldots, n-1$ where n is the number of the instances of the dataset). Then, a solution is encoded as a vector of p integer values i_j (with $j = 0, 1, \ldots, p-1$) where i_j is the index of an instance. Hence, our algorithm exploits an integer encoding. The chromosome size p is related to the reduction rate given in input. Formally, let us consider TR the dataset to be reduced characterized by n instances and $r_{\%}$ the reduction rate, the chromosome size p is as follows:

$$p = n - n * \frac{r_{\%}}{100}. \tag{4}$$

Differently from other encoding schemes for training set selection, the proposed one permits to better deal with large datasets. Indeed, the chromosome size is not equal to the

[3]https://www.w3.org/TR/2012/CR-html5-20121217/
[4]https://www.w3.org/Style/CSS/
[5]https://ecma-international.org/
[6]https://www.python.org/

TABLE I: Listing of the genetic algorithm-based wrapper TSS method implemented by TSSweb

Input: the number of the instances n of the original dataset; the number of instances to be selected p; parameters of GA (size of the population pop_size, crossover rate p_c, mutation rate p_m, tournament size t_size), termination criteria (maximum number of evaluations max_evals).

Output: the $best_chromosome$ representing the reduced dataset

```
1: gen ← 0;
2: pop ← generatePopulation(pop_size);                    // Generate randomly an initial population pop of pop_size chromosomes
3: evaluateFitness(pop);                                  // Evaluate the fitness value for each chromosome and increase the number of executed evaluations evals
4: best_chromosome ← getBestChromosome(pop);              // Select the best chromosome of the current population
5: while (evals ≤ max_evals) do
6:     offspring=executeTournament(pop, pop_size, t_size);   // Select pop_size chromosomes to compose the offspring
7:     executeSinglePointCrossover(offspring, p_c);          // Recombine chromosomes of the offspring according to a crossover rate p_c
8:     executeUniformIntegerMutation(offsping, p_m);         // Mutate chromosomes of offspring with a mutation probability p_m
9:     evaluateFitness(offspting);                           // Evaluate the fitness value for the changed chromosomes and increase the number of executed evaluations evals
10:    pop ← offspring;                                      // The new population is replaced by the offspring
11:    best_chromosome ← getBestChromosome(pop);
12:    gen ← gen + 1;                                        // Increment number of iterations
13: end while
14: return best_chromosome;
```

number of instances of the dataset but to the number of instances of the reduced dataset. However, dealing with datasets with millions of instances will require to enhance TSSweb with technologies from Big data area. As for recombination operators, we use the well-known single-point crossover and the uniform mutation. These operators are common for the designed integer encoding. Finally, as the selection mechanism, our evolutionary-based TSS method uses the tournament selection.

IV. Experiments and Results

Being a web-based tool, TSSweb enables researchers coming from different domain areas without programming skills to compute a reduced training dataset easily. However, this advantage would be not enough if the computed reduced dataset is not characterized by a good quality. Therefore, this section is devoted to show the good performance of the proposed training set selection by means of a set of experiments involving a set of well-known datasets and a comparison with two baseline approaches. Hereafter, more details about the experimental configuration and the results are given.

A. Experimental set-up

The experiments involve well-known datasets from the UCI Machine Learning Database Repository[7]. Table II shows the features in terms of number of attributes, instances and classes of the selected datasets. As it is possible to see, the datasets have been selected to cover a different set of values for the aforementioned features. Moreover, datasets belonging to classification or regression data have been selected to study all TSSweb functionalities. Each dataset has been split in training and test data. Training data is the dataset that will be reduced, whereas, test data will be used for computing the study of the performance of the proposed TSS method. The percentage of test data is the 20% of the original data.

[7]https://archive.ics.uci.edu/ml/index.php

TABLE II: Datasets information. Note that C stands for classification and R for regression.

Dataset name	#instances	#attributes	#classes	Task
bupa	345	6	2	C
glass	214	9	7	C
heart	270	13	2	C
laser	993	4	-	R
stock	950	9	-	R

The evaluation of the proposed method is performed by using several indicators. Firstly, the quality of the reduced training dataset is computed by considering the performance obtained by the selected supervised method when trained with the reduced training dataset and applied on the original training dataset. This indicator allows to understand how much good the reduced dataset represents the original one. The performance measures are the well-known accuracy in the case of the classification task and the coefficient of determination R^2 of the prediction in the case of the regression task. Formally,

$$Accuracy = \frac{c}{t} \tag{5}$$

where t is the number of new instances to be predicted and c is the number of the correctly predicted instances. Instead, the coefficient of determination R^2 can be defined as follows.

$$R^2 = 1 - \frac{\sum_{i=1}^{t}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{t}(y_i - \overline{y})^2} \tag{6}$$

where t is the number of new instances to be predicted, y_i is the i^{th} real value to be predicted, $\hat{y}_i$ is the i^{th} predicted value and $\overline{y}$ is the mean value of the variable to be predicted. The coefficient of determination ranges from $-\inf$ to 1. Negative values arise when the mean of the data provides a better fit to the outcomes than do the predicted values.

The second indicator is the performance of the supervised method in predicting instances in the test dataset. In this case, the performance of the supervised methods trained by using the reduced dataset is compared with that obtained by two

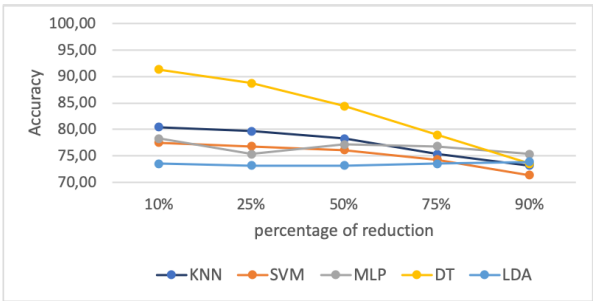
TABLE III: Experimental configuration of hyper-parameters

Algorithm	Parameters
TSS method	Population size = 100, Evaluations = 1,000, p_c = 0.8, p_m = 0.05, tournament_size=5
K-nearest neighbour (KNN)	k-value=5
Support Vector Machine (SVM)	C-value=1.0, Tolerance=0.001, maximum_number_of_iterations=200, kernel=Gaussian, gamma=1/number_of_features
Multi-layer perceptron (MLP)	number_of_hidden_layers=1, number_of_neurons=100, Tolerance=0.0001, maximum_number_of_iterations=200, learning rate=0.001
Decision Tree (DT)	criterion=gini/mse, maximum_depth=200

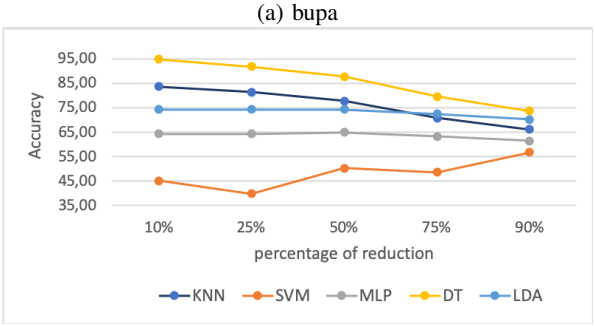
baseline approaches. The first baseline approach, denoted as *original*, consists of training the selected supervised method by taking into account all the original training dataset. The proposed TSS method will perform well if its quality measure is a bit less, equal or greater than that produced by the original dataset. Indeed, if the original dataset contain redundancy or wrong information, the computed reduction will improve the performance value, otherwise, it could lead to a slightly lower quality of the prediction. The second baseline method, denoted as *random*, consists of training the supervised method by using a dataset reduced randomly. However, to perform a fair comparison, the performance obtained by the random approach is the average of values computing using a set of randomly selected reduced datasets. The cardinality of this set is equal to the number of fitness evaluations computed by our training set selection mechanism (i.e., the number of solutions evaluated). In this case, the performance obtained by our TSS mechanism is expected to be very high with respect to the random approach. Also for this second indicator, the performance measures are the accuracy (see Eq. 5) in the case of the classification task and the coefficient of determination R^2 (see Eq. 6) of the prediction in the case of the regression task. To perform a complete experimental session, all supervised methods included in TSSweb are applied. Table III shows the experimental hyper-parameters of the our TSS mechanism and the applied supervised methods. Moreover, different values for the percentage of reduction are considered, i.e., 10%, 25%, 50%, 75% and 90%.

B. Results

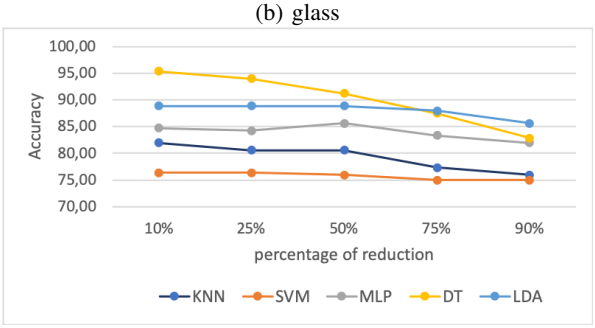
As for the first indicator of quality related to how much our TSS method succeeds to represent the original dataset, Fig. 4 displays the performance measure values (accuracy or R^2 according to the kind of task) for all datasets and for all supervised methods when trained with the TSSweb reduced datasets according to different reduction rates and applied to predict the original training data. By analysing this figure, firstly, it is possible to see a high value for the performance measures in general. That means that our TSS method performs well regardless from the supervised learning methods embedded. Quantitatively, by considering all supervised methods, all reduction rates and all datasets, the average accuracy is about 75%, whereas, the average coefficient of determination is 0.8. Typically, the performance measure value decreases when the



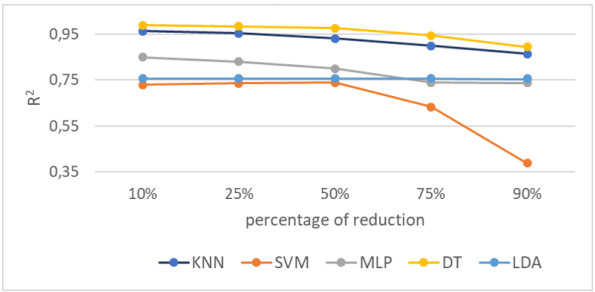
(a) bupa



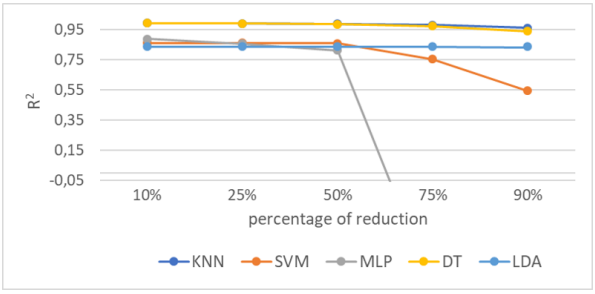
(b) glass



(c) heart



(d) laser



(e) stock

Fig. 4: Performance of the supervised learning methods trained with the reduced datasets computed by our TSS method and applied to predict the original training data.

percentage of the reduction increases. However, by computing an average value on all supervised methods and all datasets by considering the maximum reduction rate, i.e. 90%, the average accuracy is about 73%, whereas, the average coefficient of determination is 0.63. Therefore, the performance is good also when the supervised learning methods are trained with the datasets with the maximum reduction, and, this means that TSSweb succeeds to keep the most suitable instances.

As for the comparison between our TSS method and the two baseline approaches, Table IV shows the results in terms of accuracy for classification datasets and in terms of the coefficient of determination in the case of regression datasets. More precisely, the table shows the performance values of the different supervised learning methods applied on the test dataset when they are trained with the original complete datasets (the original approach), the reduced datasets computed by our TSS method and the datasets randomly reduced (the random approach). Since the performance of the our approach and the random one have been evaluated by considering different reduction rates, in order to compute the whole comparison, their reported performance values are the mean computed on the different reduction rate values. As shown in the table, the performance of the supervised methods trained with the TSSweb's reduced datasets are near or, often, greater than that obtained using the original training data. Moreover, our proposal always outperforms better than the random approach. Starting from this analysis, it is possible to say that TSSweb produces reduced datasets with a high quality.

TABLE IV: Comparison between our TSS method and baseline approaches in predicting the test dataset

Dataset bupa (C)			
Method	**Our approach**	**Original approach**	**Random approach**
KNN	65.22	60.87	62.03
SVM	71.01	72.46	64.35
MLP	71.01	68.12	66.67
DT	59.42	63.77	59.71
LDA	71.59	69.57	64.64
Dataset glass (C)			
Method	**Our approach**	**Original approach**	**Random approach**
KNN	67.91	69.77	61.40
SVM	44.65	37.21	36.74
MLP	59.07	37.21	37.21
DT	68.84	69.77	60.93
LDA	55.95	60.47	55.35
Dataset heart (C)			
Method	**Our approach**	**Original approach**	**Random approach**
KNN	66.67	61.11	60.37
SVM	63.70	59.26	58.15
MLP	80.37	72.22	55.56
DT	74.81	70.37	69.26
LDA	80.00	81.48	76.67
Dataset laser (R)			
Method	**Our approach**	**Original approach**	**Random approach**
KNN	0.78	0.80	0.77
SVM	0.61	0.66	0.55
MLP	0.62	0.70	0.53
DT	0.88	0.94	0.81
LDA	0.57	0.58	0.57
Dataset stock (R)			
Method	**Our approach**	**Original approach**	**Random approach**
KNN	0.97	0.99	0.97
SVM	0.80	0.84	0.71
MLP	0.22	0.87	-0.22
DT	0.96	0.97	0.92
LDA	0.84	0.84	0.83

V. Conclusions

The paper presents the first web-based tool, named TSSweb, for implementing the training set selection procedure. TSSweb will enable researchers coming from different backgrounds to produce good reduced datasets as shown in the experimental session. The results of the training set selection procedure are displayed in the web browser and sent via mail. In the future, TSSweb could be enhanced by implementing, in the back-end tier, several training set selection methods (e.g., fuzzy-based approaches [14]) and by allowing users to select what they want to use to produce their reduced training datasets.

References

[1] N. Verbiest, J. Derrac, C. Cornelis, S. García, and F. Herrera, "Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis," *Applied Soft Computing*, vol. 38, pp. 10 – 22, 2016.

[2] S. García, A. Fernández, and F. Herrera, "Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems," *Applied Soft Computing*, vol. 9, no. 4, pp. 1304 – 1314, 2009.

[3] G. Acampora, F. Herrera, G. Tortora, and A. Vitiello, "A multi-objective evolutionary approach to training set selection for support vector machine," *Knowledge-Based Systems*, vol. 147, pp. 94 – 108, 2018.

[4] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2003.

[5] R. Jensen and C. Cornelis, "Fuzzy-rough instance selection," in *International Conference on Fuzzy Systems*. IEEE, 2010, pp. 1–7.

[6] I. Triguero, S. González, J. M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera Triguero *et al.*, "Keel 3.0: an open source software for multi-stage analysis in data mining," 2017.

[7] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[8] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[9] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, March 2012.

[10] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693–2709, 2008.

[11] G. Acampora, G. Tortora, and A. Vitiello, "Applying spea2 to prototype selection for nearest neighbor classification," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 924–003 929.

[12] G. Acampora, P. Avella, V. Loia, S. Salerno, and A. Vitiello, "Improving ontology alignment through memetic algorithms," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, June 2011, pp. 1783–1790.

[13] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[14] J. Alcalá-Fdez and J. M. Alonso, "A survey of fuzzy systems software: Taxonomy, current research trends, and prospects," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 1, pp. 40–56, 2015.