Stacking Strong Ensembles of Classifiers

Stamatios-Aggelos N. Alexandropoulos, Christos Aridas, Sotiris Kotsiantis, Michael N. Vrahatis

▶ **To cite this version:**

Stamatios-Aggelos N. Alexandropoulos, Christos Aridas, Sotiris Kotsiantis, Michael N. Vrahatis. Stacking Strong Ensembles of Classifiers. 15th IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI), May 2019, Hersonissos, Greece. pp.545-556, 10.1007/978-3-030-19823-7_46 . hal-02331304

HAL Id: hal-02331304
https://inria.hal.science/hal-02331304v1

Submitted on 24 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Stacking strong ensembles of classifiers*

Stamatios-Aggelos N. Alexandropoulos[1], Christos K. Aridas[1], Sotiris B. Kotsiantis[1], and Michael N. Vrahatis[1]

Computational Intelligence Laboratory (CILab)
Department of Mathematics
University of Patras
GR-26110 Patras, Greece
http://cilab.math.upatras.gr/
alekst@math.upatras.gr, char@upatras.gr,{sotos,vrahatis}@math.upatras.gr

Abstract. A variety of methods have been developed in order to tackle a classification problem in the field of decision support systems. A hybrid prediction scheme which combines several classifiers, rather than selecting a single robust method, is a good alternative solution. In order to address this issue, we have provided an ensemble of classifiers to create a hybrid decision support system. This method based on stacking variant methodology that combines strong ensembles to make predictions. The presented hybrid method has been compared with other known-ensembles. The experiments conducted on several standard benchmark datasets showed that the proposed scheme gives promising results in terms of accuracy in most of the cases.

Keywords: Decision support systems · Supervised machine learning · Ensembles of classifiers

1 Introduction

An *ensemble* of classifiers [16] combines the decision of individual classifiers in some way in order to classify new instances. A various number of methods have been presented for the creation of such a set of classifiers [4]. The most common and widely used ways to create such a combination of classifiers are the following three:

(a) Using an individual learning method and different subsets of training data.
(b) Using an individual training algorithm and various training parameters.
(c) Using different learning methods.

One can reasonably be asked why an ensemble of classifiers may provide better results than a single classifier. The reasons are both representational, statistical and computational and are presented in [4]. The training data cannot produce informative knowledge in some of the cases in order to select a single

* Supported by Hellenic State Scholarships Foundation (IKY).

classifier. This may be a possible cause of the aforementioned issue since the training data is too small concerning the amount of the hypothesis space.

Despite the fact that the interest gathered about methods of ensembles in recent years is so big, further study is required since it is not clear which method is the best. In the present work, we have performed a hybrid scheme that combines the following ensemble methods: *Random Forest* [2], *ExtraTrees* [10] and *Gradient Boosting* [9] algorithm using a stacking variant methodology. The proposed hybrid decision support system is compared with several ensembles methods on a variety of benchmark datasets. The results obtained showed that the provided method has better accuracy in most of the problems.

The rest of the paper is organized as follows. In the next section, we present well-known methods for creating ensembles of classifiers, while in Section 3 we consider the proposed ensemble method. The conducted experiments and the comparisons with other ensemble methods are presented in Section 4. Finally, we conclude in Section 5 with a summary and further research remarks.

2 Well-known methods for creating ensembles

There are a plethora of methods using a single classifier in order to solve a prediction problem. However, the need for models that will make a more accurate prediction is intense. This purpose, as mentioned in the introduction of our work, is served by a combination of classifiers. In order to create a good ensemble of classifiers, there are different methodologies. Each of these can provide a varied rate of correctly classified instances. Thus, the overall goal of the methods belonging in decision support systems is to achieve reliable and very accurate prediction results. This section presents widely used methods for building ensembles.

Bagging technique or *Bootstrap aggregating* [1] is one of the most known and widely used technique for creating ensembles of classifiers. A key feature of this method is the training of each learner using different training sets. Specifically, let N be a set of training data of size n. *Bagging* method selects from this set n' random examples, creating a set of training subsets, let N_i. The selection of instances is done using a distribution, for example the uniform distribution, while the examples are drawn with replacement. Since the building of the subsets is done in that way, it is possible some of the examples to be repeated in N_i and some duplicates and omissions to contained in regard to the starting training set N. This execution produces a set of classifiers. Then, a voting process is taking place in relation to the predictions of each classifier. The result of the vote gives the final decision.

A well-known meta-algorithm is also the *Boosting* approach [8]. One could say that this method resembles with the bagging approach since several sub-samples are used for training a single learner. Moreover, this algorithm was based on the question "Can a set of weak learners create a single strong learner?" [14]. *Boosting* method is directly related to this question as it focuses on the performance of a weak classifier in order to enhance it, especially in cases where instances are not

effectively been learned. Thus, rather than selecting the instances in a random way based on a distribution, the algorithm focuses on the training instances that have not been learned correctly. In order to create a strong classifier, after several steps, the prediction is done through a weighted voting procedure. In detail, each classifier prediction takes a weight according to the classification accuracy that it achieved.

Besides the aforementioned techniques that mostly train classifiers in a subset of training data, there are methods that create a model using a set of learning algorithms on the entire training dataset. Then, by choosing an appropriate voting pattern, they combine the responses in order to obtain the final prediction. Through this process, different learning algorithms are used to increase the diversity of prediction errors of the models. This is because they differ in both the search process and the representation. In conclusion, the models that created are more likely to make mistakes at different points, as different learning biases are used. Finally, utilizing the majority of forecasts we easily end up in the final decision, as no prior training is required.

A well-known method that uses the above-mentioned procedure is *stacked generalization* approach or *Stacking* [26]. In this method, the production of a strong, high-level learner with high generalized performance is the main goal. To obtain this, a set of different classifiers are combined. To achieve an appropriate combination of the base predictions, a learning algorithm is used. In order to perform the final forecast, the model is utilizing the following steps: (a) *Level zero data*. All the base learners run on the original dataset, (b) *Level one data*. After the zero level, the predictions made by the classifiers are considered as new data and (c) *Final prediction*. Another learning process takes place using the level one data as new inputs and as output, the final prediction is gained.

An alternative idea was used by [25]. The proposed method is an extension of the well-known stacking [26] and is based on class probability distributions rather than the predictions. Thus, each learner does not present a final prediction for the class to which an example belongs. As a result, it gives an estimated probability for all classes. The authors, in order to achieve meta-level learning, used the *Multi-response Linear Regression (MLR)*.

Several approaches have been provided in order to improve the performance of [25]. In particular, in [7] adopted the same strategy except for the meta-learning procedure. The authors, used model tree induction instead of *MLR* and the obtained results showed that better performance is achieved. Furthermore, another modification is proposed by [21]. In this method, instead of the overall class probability distributions, only class probabilities concerning the true class are taking into account. The experimental results showed that by this variation the accuracy of this method seems to be improved.

In [15] another stacking algorithm, called *Troika*, is presented in order to tackle multi-class classification tasks. Specifically, the proposed method instead of one meta-learner uses three layers of combining classifiers: (a) The zero layer consists of base learners, (b) The first one contains specialist classifiers, (c) The second layer is composed by meta-learners and (d) In the last layer is met the

super-classifier which take the predictions of the previous layer in order to make the final prediction of the model. The main comparison included the *Troika* method, the basic stacking scheme and an improvement of standard *Stacking*, the *StackingC* method. The experiments conducted showed that the presented method outperforms the other two methods in terms of accuracy irrespective of the base-classifier binarization method that was selected.

Stacking ensemble methodology can be seen as a combinatorial optimization problem. This is because both base classifiers and meta-classifier have to be determined in an optimal way. In [3] an adaptive metaheuristic search method was adopted in order to create a new stacking approach, called *Ant Colony Optimization (ACO) Stacking* algorithm. Thus, given a set of base learners, let B, which consists of n classifiers, ACO algorithm [5] determines a subset of B consisting of a smaller amount of learners. This set of classifiers is expected to has better classification accuracy than the original. The authors compared *ACO-Stacking* approach with well-known ensembles such as *Bagging*, *AdaBoost*, *StackingC* and *Random Forest*. The experimental results showed that the performance of *ACO-Stacking* algorithm is promising.

A similar approach in which the problem of selecting the appropriate configuration of base learners and the meta-classifier is proposed in [23]. The main difference relating to [3] concerns the swarm intelligent search algorithm that used for the optimization task. Particularly, the authors used the well-known *Artificial Bee Colony (ABC)* algorithm [13] in order to build two different types of stacking. Thus, the optimal configuration of base learners is taking place in the beginning. In that phase, the meta-learner that considered is a fixed learning algorithm. During the meta-phase, the ABC algorithm optimize at the same time the base learners and the meta-learner too. The *ABC-Stacking* approach was tested in several well-known benchmark datasets and compared with different predictive learners, basic ensemble techniques, such as *GA-Stacking* [17] and *ACO-Stacking*. The experiments conducted by the authors showed that the provided approach has addressed effectively the configuration of base-classifiers and the selection of meta-learner too. Moreover, the performance of *ABC-Stacking* is comparable to the best results cited so far.

In [19] a strategy which is similar to the *Stacking* methodology is adopted in order to apply a sequential learning algorithm in multi-class problems. For a more extensive study of *Stacking* methodology, *Stacking* methods and related with *Stacking* approaches that have been developed over the last 20 years the reader is referred to [22].

3 Proposed hybrid scheme

As with any classifier, adding new input features can improve classification accuracy when the new features contain new information about the class. This performance improvement isn't guaranteed because classifiers are imperfect and may not be able to exploit the information. If the new features share information with existing features, the new features may or may not help. In all cases, the

curse of dimensionality must be considered. The presence of more features can hurt many classifiers and can increase opportunities for overfitting. The situation is similar when adding new base classifiers to a stacking setup, because the base classifiers' outputs are features for the final classifier. All the same arguments from above hold here. In this case, these 'second level' features are likely correlated because all base classifiers are all trying to predict the same case. However, they do it suboptimally. The hope is that they behave in different ways, so that the final classifier can combine the noisy predictions into a better final prediction. Loosely, then, adding new base classifiers has the best chance of helping when they do a good job and behave differently than existing base classifiers, but this isn't guaranteed. In all cases, the time of training must be considered. In our case, we select to use three strong ensemble methods as base learners: *Random Forest*, *ExtraTrees* and *Gradient Boosting*.

According to Brieman's definition in [2], a *Random Forest* is a classifier that consists of a set of tree-structured classifiers. A worth point is that in each tree the decisions are made by a random way when splitting a node, without taking all the features into account. In this process, each tree is trained in a sub-sample in a similar way with bagging and votes for the most popular class concerning a given input. This kind of decision trees using averaging can improve the prediction accuracy. In addition, can achieve the control of the overfitting. As in the *Bootstrap aggregating* the creation of the sub-samples is done with replacement and the size of each sub-sample is equal to the beginning sample.

A similar method to the Random Forests is the *Extremely randomized Trees* or *ExtraTrees* [10]. This method is successfully applied and produces good results in terms of accuracy while effectively controls the overfitting issue. The difference with the Random Forests can be found in two points: (a) The training of each tree is made without using a bootstrap sample but a whole training set and (b) The cut-point for splitting a node is selected by a fully random way.

Gradient Boosting [9] is a widely used approach for handling regression and classification tasks, which is mainly based on decision trees classifiers. The main strategy of this method is similar to other boosting methods (such as *AdaBoost* or *LogitBoost*). In a first phase build a set of weak learners in order to make an initiate prediction. In a second phase optimize the prediction accuracy of the model based on the minimization of a cost function. Specifically, gradient boosting iteratively determining a function that points in the negative gradient direction.

In our stacking algorithm, the set of meta-data will consist of the predictions given by the base ensembles and the true class for each training case. Thus, the classifier trained on this dataset is the so-called meta-learner. Our approach uses the class probabilities correlated with the true class rather than all class probabilities as in the standard *Stacking*.

Another point that is worth mention is the learning process. The goal is to build a fast learning procedure. This is achieved by reducing the dimensionality of meta-data set by a factor associated with the number of classes. Despite this reduction, the accuracy for the two-class problems remains unaffected. More-

over, the learning algorithm that we use at the meta-level is the well-known *Logistic Model Tree* classifier [24]. This is because piecewise linear approximations are gained through these models. The figure 1 illustrates the process that was adopted in order to create a strong classifier, while the proposed method is illustrated in the Algorithm 1.

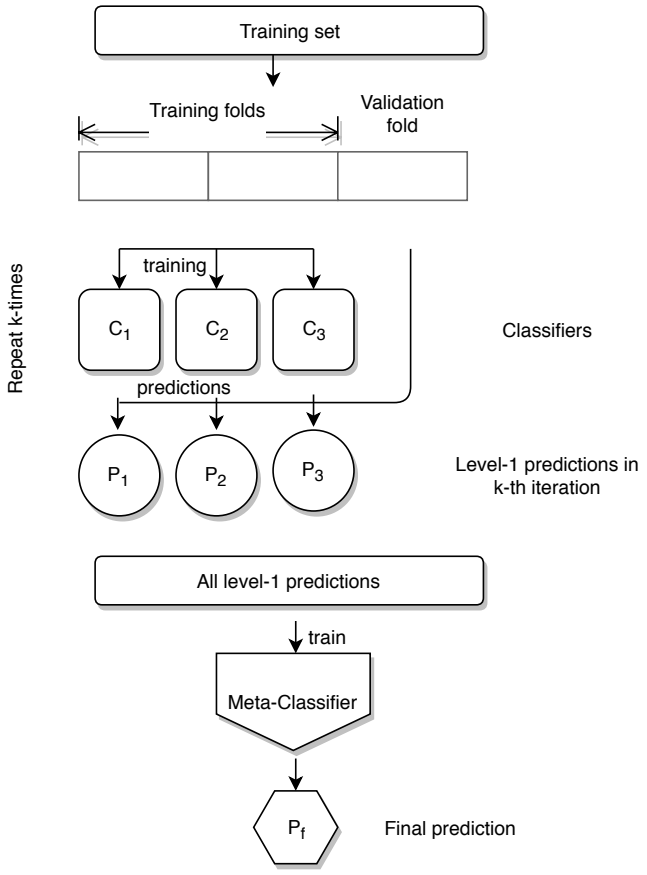

Fig. 1. Stacking strong ensembles process

Algorithm 1 Stacking with k-fold cross validation

Input Training data $S = \{x_i, y_i\}_{i=1}^{m}$, $x_i \in \mathcal{R}^n, y_i \in \mathcal{C}$, where $\mathcal{C}$ denotes the classes.
Output A stacking meta-classifier ensemble P_f
Step 1 Adopt cross validation approach in preparing a training set for second-level *Logistic Model Tree* classifier
Split the starting dataset into $k(=3)$ equal-size subsets $\{S_1, S_2, S_3\}$ in a random way
for $j \leftarrow 1$ to k **do**
  Learn first-level classifiers
  for $t \leftarrow 1$ to 3 **do**
    Train a learner l_{kt} from $S \backslash S_k$
  end for
  Construct a training set for second-level model tree classifier
  for $x_i \in S_k$ **do**
    Get a record $\{x'_i, y_i\}$, where $x'_i = \{p_{k1}(x_i), p_{k2}(x_i), p_{k3}(x_i)\}$
  end for
end for
Step 2 Learn a second-level classifier
Learn a new classifier p' from the collection of $\{x'_i, y_i\}$
Step 3 Re-learn first-level learners
for $t \leftarrow 1$ to 3 **do**
  Learn a classifier p_t based on S
end for
Return $P_f(x)$

4 Comparisons and Experimental results

For our experiments, well-known datasets from UCI Machine Learning repository [6] were used. Specifically, 38 problems from different fields were selected, which differ in both the number of their attributes and their classes. In Table 1, there is a brief description of these datasets such as the number of features, the number of instances and the number of output classes.

The accuracy of the classifiers was evaluated according to the following procedure: The training dataset was divided into ten equally-sized subsets, and for each of them, the classifier was trained in the remainder subsets. Then, for each method, we run the five fold cross-validation procedure and the average value of five fold cross validations was measured. The experiments have been conducted with Python using the available implementations from the *scikit-learn* [18] and *MLxtend* library [20]. In Table 2 the obtained results for our hybrid scheme and for methods which participate in the comparison are exhibited. The best performing scheme for each dataset is described using boldface writing. It can be easily seen that the proposed stacking ensemble method outperforms the other three well-known ensembles in most of the cases and thus we can say that the proposed method is more robust.

Furthermore, statistical tests have been performed in order to check the significance of the results. In particular, due to the small number of comparison

Table 1. Collection of 38 multi-class datasets from the UCI Machine Learning Repository. The number of instances, the number of features as well as the number of classes, are exhibited.

Dataset	#Instances	#Features	#Classes
audiology	226	69	24
autos	205	25	7
badges	294	11	2
balance-scale	625	4	3
breast-cancer	286	9	2
wisconsin-breast-cancer	699	9	2
horse-colic	368	28	2
credit-rating	690	15	2
german-credit	1000	20	2
pima-diabetes	768	8	2
glass	214	9	6
grub-damage	100	11	2
haberman	306	3	2
hear-cleveland	303	13	5
heart-hungarian	294	12	2
heart-statlog	270	13	2
hepatitis	155	19	2
hypothyroid	3772	30	2
ionosphere	351	34	2
iris	150	4	3
labor	57	16	2
lymphography	148	18	4
monk1	556	7	2
monk2	601	7	2
monk3	554	7	2
mushroom	8124	22	2
primary-tumor	339	17	21
students	344	11	2
sick	3772	29	2
sonar	208	60	2
soybean	683	35	19
relation	2201	3	2
vehicle	846	18	4
vote	435	16	2
vowel	990	13	11
waveform	5000	40	3
wine	179	13	3
zoo	101	17	7

methods, the non-parametric Friedman test [11] has been conducted. Therefore, in Table 3 the Friedman test ranking is presented. Moreover, p-value in all the comparisons indicates that the null hypotheses should be rejected. Thus,

Table 2. Classification accuracy and standard deviation of the compared methods, where "ET" denotes ExtraTrees, "GB" represents Gradient Boosting, while "RF" denotes Random Forest.

Dataset	Stacking	ET	GB	RF
audiology	79.13±5.01	66.13±7.28	**82.66±5.28**	76.99±5.79
autos	82.15±6.78	68.49±6.22	**82.54±5.30**	79.12±4.17
badges	98.98±3.28	96.86±4.30	**100.00±0.00**	99.11±1.92
balance-scale	**89.38±1.30**	78.78±2.93	87.33±2.01	81.66±2.41
breast-cancer	70.55±4.20	64.07±6.71	69.58±5.87	**71.82±3.81**
wisconsin-breast-cancer	**96.42±1.71**	94.56±2.00	96.17±1.74	95.91±1.83
horse-colic	83.64±3.51	73.97±5.20	**83.80±2.99**	83.32±3.53
credit-rating	**86.70±1.97**	79.19±4.01	86.14±1.95	86.09±2.13
german-credit	**75.92±2.53**	66.70±2.79	75.52±2.40	74.10±2.70
pima-diabetes	**75.89±3.52**	67.89±2.79	75.73±3.63	74.04±2.95
glass	**76.27±5.63**	63.65±8.37	75.34±4.85	73.10±5.53
grub-damage	**41.81±6.68**	40.13±6.39	38.97±7.73	41.29±7.90
haberman	**72.29±2.53**	65.17±4.87	69.60±3.84	70.52±3.16
hear-cleveland	80.34±5.07	76.38±4.63	**80.80±4.66**	79.81±5.60
heart-hungarian	81.64±4.49	75.92±4.83	**82.25±5.04**	80.68±5.73
heart-statlog	**80.96±4.43**	74.22±5.37	78.89±4.66	79.70±4.19
hepatitis	**83.10±5.98**	78.19±7.41	81.94±5.59	81.94±6.45
hypothyroid	**99.49±0.29**	94.18±1.02	99.46±0.31	99.14±0.32
ionosphere	**93.28±3.13**	87.12±4.03	92.88±2.81	92.31±2.64
iris	94.27±3.79	91.73±5.54	**95.33±3.43**	**95.33±3.47**
labor	90.58±9.97	85.48±12.86	85.73±10.66	**90.91±9.62**
lymphography	83.79±7.67	77.31±9.13	**85.15±7.13**	82.42±7.07
monk1	98.71±3.03	74.69±12.95	**98.87±2.75**	86.15±6.52
monk2	63.46±6.91	54.80±7.42	**64.74±8.42**	57.50±6.43
monk3	**91.65±4.01**	82.69±7.32	91.17±4.20	90.66±5.19
mushroom	**100.00±0.00**	99.99±0.02	99.99±0.05	**100.00±0.00**
primary-tumor	**42.48±3.29**	34.28±4.69	39.82±4.68	40.81±4.22
students	83.96±3.75	80.46±3.37	**84.19±3.73**	82.62±3.30
sick	98.65±0.34	94.59±1.05	**98.68±0.31**	98.26±0.42
sonar	**83.38±6.26**	69.80±5.79	82.51±6.14	79.46±7.22
soybean	93.00±2.04	88.20±3.13	**93.06±1.99**	92.94±1.79
relation	**79.02±0.94**	78.94±1.02	78.94±1.02	78.95±1.05
vehicle	**75.44±2.41**	65.91±2.92	75.42±2.29	74.02±2.64
vote	**96.00±2.13**	92.92±2.96	95.31±2.32	95.49±2.87
vowel	**92.93±1.93**	75.82±3.31	89.78±2.09	91.37±2.03
waveform	**85.61±1.21**	68.37±1.52	85.52±1.20	81.36±1.22
wine	**96.73±2.77**	87.97±6.14	95.05±4.58	96.18±3.61
zoo	93.65±4.70	88.70±7.69	89.30±4.46	**94.89±4.09**

there are methods whose performance difference was statistically significant to the others. In Table 4, the obtained results of the post-hoc Holm test [12] are exhibited.

Table 3. Rankings of the algorithms using the Friedman test

Stacking	1.48684
Gradient Boosting	2.05263
Random Forest	2.51316
ExtraTree	3.94737

Table 4. Post-hoc Holm test using Stacking as control method

Comparison	Statistic	Adjusted p-value	Result
Stacking vs ExtraTree	8.30769	0.00000	H_0 is rejected
Stacking vs Random Forest	3.46524	0.00106	H_0 is rejected
Stacking vs Gradient Boosting	1.91033	0.04609	H_0 is rejected

5 Conclusions

Hybrid decision schemes seem to be a reliable and effective option in order to tackle decision-making tasks in several scientific fields. Compared to the accuracy that a single classifier can achieve, an ensemble method seems to be able to reach better classification results. However, designing such a method is a difficult problem, as we have a plethora of basic classifiers for use. Stacking methodology is an appropriate and accurate procedure in order to create a strong combination using the best over-all stacked classifier. Despite the fact that a variety of ensemble methods have been designed, creating the best combination for solving specific classification problems remains a hot and ongoing project. In this paper, proposed a Stacking strong ensemble methodology using *Logistic Model Trees* and three well-known ensembles. The experimental results conducted over well-known datasets showed that the provided scheme gives promising results in most of the benchmark problems. Nevertheless, there are several issues that need further study, such as the rules that build a suitable set of classifiers with the highest possible accuracy. Moreover, further comparisons of our method with well-known variations of Stacking such as [3, 17] and implementation in regression problems could be an interesting task for future research.

6 Acknowledgements

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

References

1. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
2. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
3. Chen, Y., Wong, M.L.: An ant colony optimization approach for stacking ensemble. In: Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on. pp. 146–151. IEEE (2010)
4. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)
5. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). vol. 2, pp. 1470–1477. IEEE (1999)
6. Dua, D., Taniskidou, E.K.: Uci machine learning repository [http://archive. ics. uci. edu/ml]. university of california, school of information and computer science. Irvine, CA **144** (2017)
7. Džeroski, S., Ženko, B.: Is combining classifiers with stacking better than selecting the best one? Machine learning **54**(3), 255–273 (2004)
8. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: Icml. vol. 96, pp. 148–156. Citeseer (1996)
9. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
10. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine learning **63**(1), 3–42 (2006)
11. Hodges, J., Lehmann, E.L., et al.: Rank methods for combination of independent experiments in analysis of variance. The Annals of Mathematical Statistics **33**(2), 482–497 (1962)
12. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian journal of statistics pp. 65–70 (1979)
13. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer âĂę (2005)
14. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. Journal of the ACM (JACM) **41**(1), 67–95 (1994)
15. Menahem, E., Rokach, L., Elovici, Y.: Troika–an improved stacking schema for classification tasks. Information Sciences **179**(24), 4097–4122 (2009)
16. Murty, M.N., Devi, V.S.: Combination of classifiers. In: Pattern Recognition, pp. 188–206. Springer (2011)
17. Ordóñez, F.J., Ledezma, A., Sanchis, A.: Genetic approach for optimizing ensembles of classifiers. In: FLAIRS conference. pp. 89–94 (2008)

18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
19. Puertas, E., Escalera, S., Pujol, O.: Generalized multi-scale stacked sequential learning for multi-class classification. Pattern Analysis and Applications **18**(2), 247–261 (2015)
20. Raschka, S.: Mlxtend: Providing machine learning and data science utilities and extensions to pythonâĂŹs scientific computing stack. The Journal of Open Source Software **3**(24) (2018)
21. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: Proceedings of the nineteenth international conference on machine learning. pp. 554–561. Morgan Kaufmann Publishers Inc. (2002)
22. Sesmero, M.P., Ledezma, A.I., Sanchis, A.: Generating ensembles of heterogeneous classifiers using stacked generalization. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **5**(1), 21–34 (2015)
23. Shunmugapriya, P., Kanmani, S.: Optimization of stacking ensemble configurations through artificial bee colony algorithm. Swarm and Evolutionary Computation **12**, 24–32 (2013)
24. Sumner, M., Frank, E., Hall, M.: Speeding up logistic model tree induction. In: European Conference on Principles of Data Mining and Knowledge Discovery. pp. 675–683. Springer (2005)
25. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of artificial intelligence research **10**, 271–289 (1999)
26. Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)