Classification and Metaclassification in Large Scale Data Mining Application for Estimation of Software Projects

Dorota Dzega
Faculty of Economics and
Information Technology
West Pomeranian Business School
Zolnierska 53, 71-210 Szczecin, Poland
Email: ddzega@zpsb.szczecin.pl

Wieslaw Pietruszkiewicz
Faculty of Computer Science and Information Technology
West Pomeranian University of Technology in Szczecin
Zolnierska 49
71-210 Szczecin, Poland
Email: wpietruszkiewicz@wi.zut.edu.pl

***Abstract*—In this article we present an application of Artificial Intelligence for estimation of software projects. The research presented herein was based on several methods of classification and metaclassification. Due to increasing significance of Open Source, we have selected projects being hosted on the leading platform for Open Source projects – Sourceforge.net.**

In the first part of article, we describe steps of data extraction which was a large scale task because the datasource contained tens of tables and hundreds of fields, that were originally gathered to be used by project management web-based system. Therefore extraction of meaningful data required analysis of databases structure and transformation of sets of records into a four datasets. These datasets were used to predict four factors important to project management i.e skills, time, costs an effectiveness. Later, we present the results of experiments, that were performed using C4.5, RandomTree and CART algorithms. In the final part of this article, we describe how boosting and bagging metaclassifiers were applied to improve the results and we also analyse influence of their parameters on generalization abilities an prediction accuracy.

***Index Terms*—Classification, Metaclassification, Decision trees, Software estimation, Project management**

I. INTRODUCTION

Currently many popular software applications are being developed as Free Libre/Open Source Software (OS later in this article). The results of these projects, done by team members cooperating via web–systems, sometimes overperform proprietary software. Therefore OS projects become strong competitors to the classic closed-source software products. As the result, methods of software management must be extended by solutions specially tailored to OS characteristic.

The basic assumption of project management is that the knowledge and experience, acquired from the past projects, help to effectively manage risk during software development. It is coherent with the basic purpose of data mining, that focuses on knowledge extraction from the past observations and its conversion to forms easily understandable and usable in the future e.g. rules or charts.

Herein, we present an large scale Data Mining application, whose aim was to create a set of models supporting decision making by providing prognosis of the most important features of software project. All data used in presented experiments were extracted from SourceForge.net being the leading OS hosting platform. The number of existing tables and the number of stored records were a serious test to capabilities of prediction methods, that applied to this complicated real life problem must have proven their speed, precision and low memory requirements. Moreover, the methods of machine learning used herein were carefully selected after analysing of their ability to work with datasets containing large number of unordered, non-numeric attributes.

We decided to examine OS project as a fast growing part of software projects, therefore an AI–based project management is highly desired for these projects. As the used dataset was large, due to number of records and many various attributes, we used it as a real–life test for different AI methods.

II. DATASETS

The source of data about OS projects was “A Repository of Free/Libre/Open Source Software Research Data” being a copy of internal databases gathered by SourceForge.net platform [1]. In presented research a potential source of data (see complexity of database on Figure 1) contained a large number of features and its form of storage was not designed to be later used in knowledge extraction process. It was built to store all data necessary to run a web-based project management service offering forum, subversion control or tasks assignments. This caused that the presented problem was a modelling example of real life data mining application i.e. situation where meaningful attributes must be extracted from data repositories and the scale causes that some algorithms fail. It must be noted that sometimes methods that work fine during purely scientific experiments are useless in large scale problems due to high demand on resources or low speed. Technically, the data source contained almost 100 tables and a monthly increase of data was estimated at 25 GB.

This data repository was previously a subject of research e.g. [2] examined logistic regression, decision trees and neural net-

works in analysis of success factors for Open Source Software. The other research [3] examined similarity measures for OSS projects and performed clustering, while research [4] described how abandonment of an OS project could be predicted. Unlike the previous researches, presenting selected aspects of data mining applied to OSS, we present herein a complete analysis of 4 projects dimensions. The described research was based on an assumption that it is more important to offer a prediction of a few factors, that will aid a successful project management, than to predict if project become successful or not (like in some previous researches).

The extracted dataset contained information about the most important aspects of OS projects. These four datasets were [5]:

- *project scope* Z_t - the duration of project from the moment of project initialization (project registration) till the last published presentation of the project effects; containing 39 attributes, including 8 attributes pertaining to the project field (d_p), 28 attributes pertaining to the project resources (z_p) and 3 attributes pertaining to project communication (k_p),
- *project time* C_t - the time of task completion expressed in working hours spent on completing a particular task; containing 12 attributes, including 7 attributes pertaining to general conditions of task completion (w_t) and 5 attributes pertaining to the resources of persons completing the task (z_t),
- *project cost* K_t - the average number of working hours spent by a particular project contractor on task completion; containing 18 attributes, including 8 attributes pertaining to the participant competence (z_u) and 10 attributes pertaining to the participant activity (a_u),
- *project effects* E_t - the number of completed tasks as of the date of diagnosis; containing 21 attributes, including 16 attributes pertaining to activity of project execution (r_a) and 5 attributes pertaining to communication activity related to project execution (k_a).

The numeric characteristic of created dataset was presented in Table I. The column *Reduced records* denotes how many records passed through filters e.g. we have excluded empty projects or projects with an empty development team.

To select the most important attributes we have used *Information Gain Ratio*, that in its core calculates change of entropy from state X to $X|A$ (information gain caused by feature A). Assuming that $H(..)$ is an entropy function, the information gain may be calculated as $IG(X, A) = H(X) - H(X|A)$. Selecting a subset of attributes helped to limit space decisions and increase classification accuracy by rejecting unnecessary features, being informational noise. For each dataset we have examined various models, where number of input was changing from 1 to D_i, where D_i was a number of attributes for i dataset. Figure 2 presents an example of this step of experiments for *Time* dataset.

The selected sets of information attributes for the best prediction models (desired attributes) were:

$$\begin{aligned} Z_t &= \{z_1, d_8, d_1, z_4, d_7, z_8, z_2, d_5, d_3\}, \\ C_t &= \{w_1, w_7, w_5, w_2, z_5, w_4, z_4, z_3, z_2\}, \\ K_t &= \{z_1, a_1, z_6, z_8, a_2, a_4\}, \\ E_t &= \{r_2, r_{13}, r_{11}, r_{14}, r_{10}, r_9, r_8, r_{12}, k_5, k_4, r_7, k_1\}. \end{aligned}$$

We found that number of features may be reduced more, as there existed smaller subsets for which prediction accuracy was not significantly worse than maximum. The selected smaller subsets of information attributes, required for the prediction of project features were:

$$\begin{aligned} Z_t &= \{z_1, d_8, d_1, z_4\}, \\ C_t &= \{w_1, w_7, w_5, w_2\}, \\ K_t &= \{z_1, a_1\}, \\ E_t &= \{r_2, r_{13}, r_{11}\}. \end{aligned}$$

Analysing the results of this stage, we prepared an explanation of individual information attributes for *project scope* Z_t:

- the average number of working hours spent on completion of the project tasks (z_1),
- number of selected project tasks (d_8),
- subject scope of the software (d_1),
- number of project contractors (z_4),
- number of selected subprojects (d_7),
- number of project contractors at the position of a Developer (z_8),
- number of unique time zones (z_2),
- software language (d_5),
- user interface (d_3).

For *project time* dimension selected attributes were C_t:

- task completion time, expressed as a number of days from the task initialization till its completion (w_1),
- number of tasks within a subproject (w_7),
- number of preceding tasks which a particular task depends on (w_5),
- percentage of task completion (w_2),
- skills of a project contractor completing the task (z_5),
- task status (w_4),
- role / position of a project contractor completing the task (z_4),
- role / position of a project contractor assigning the task (z_3),
- number of contractors assigned to complete the task (z_2).

The *project cost* could be predicted by K_t:

- registered time of the project contractor (mths) (z_1),
- number of assigned tasks (a_1),
- number of unique skills of project contractor (z_6),
- the most frequently held role/ assigned position (z_8),
- number of projects executed by a particular contractor (a_2),
- number of posts sent by a particular contractors to different project - related Web forums (a_4).

The last dimension *project effects* contained attributes E_t:

- the average task completion status in percentage (r_2),
- number of tests in CVS version control system (r_{13}),
- number of open artefacts (additional artefacts in the project) (r_{11}),

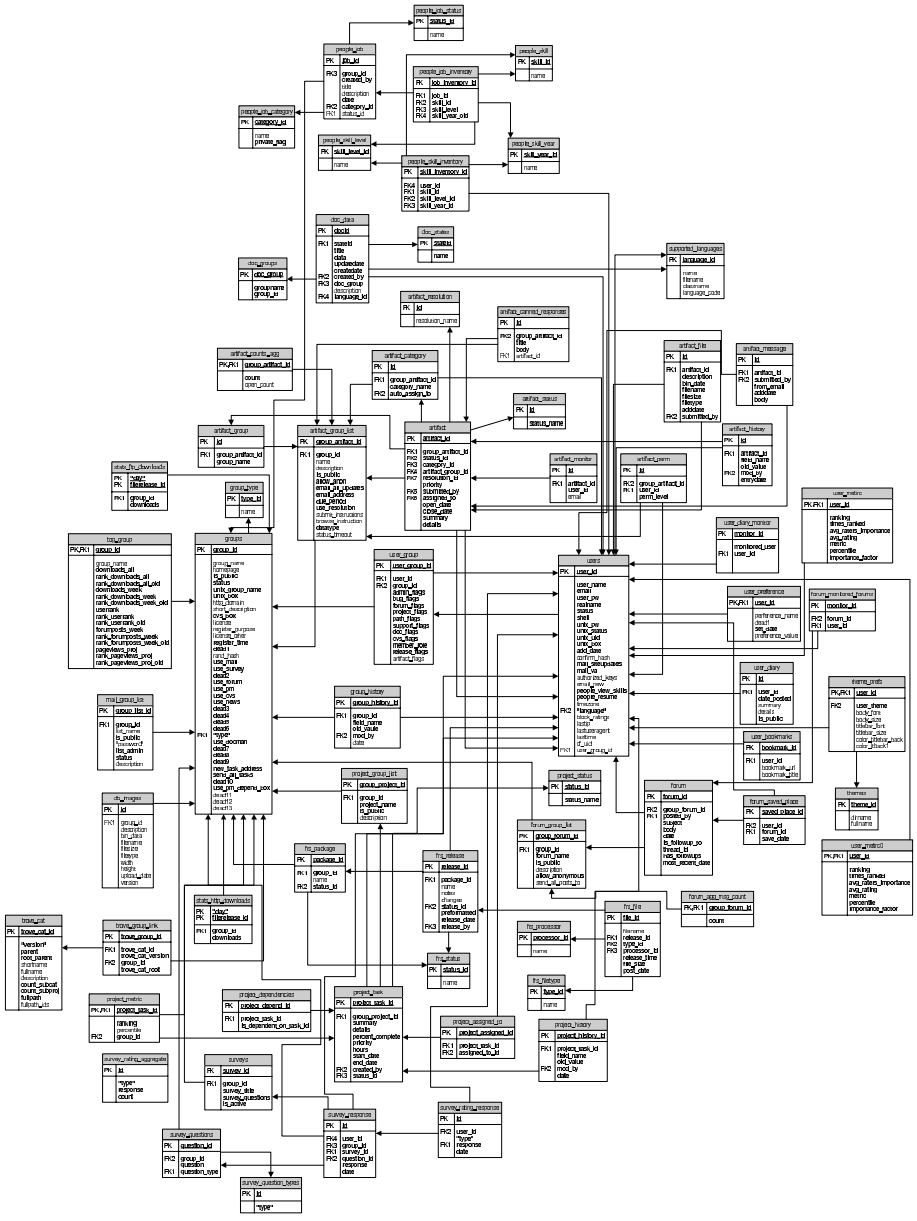

Fig. 1. Structure of tables used by SourceForge.net project management system [1]

TABLE I
DETAILS FOR *Scope*, *Time*, *Costs* AND *Effects* DATASETS.

Dataset	*Unique records*	*Reduced records*	*Objects*	*Attributes*
Scope	167698	167698	2881	39
Time	233139	104912	77592	12
Costs	127208	20353	10889	18
Effects	96830	15492	64960	21

- number of comments in CVS version control system (r_{14}),
- number of closed error reports (r_{10}),
- number of open error reports (r_9),
- number of requests for help (r_8),
- number of closed artefacts (additional artefacts in the project) (r_{12}),
- number of project mailing lists (k_5),
- number of threads on forums assigned to particular project (k_4),
- number of closed binary code modifications (r_7),
- number of project documentation groups (k_1).

These attributes were used in next stage of experiments. It must be noticed, that some of them to be created were calculated from other database fields or were a result of frequency analysis.

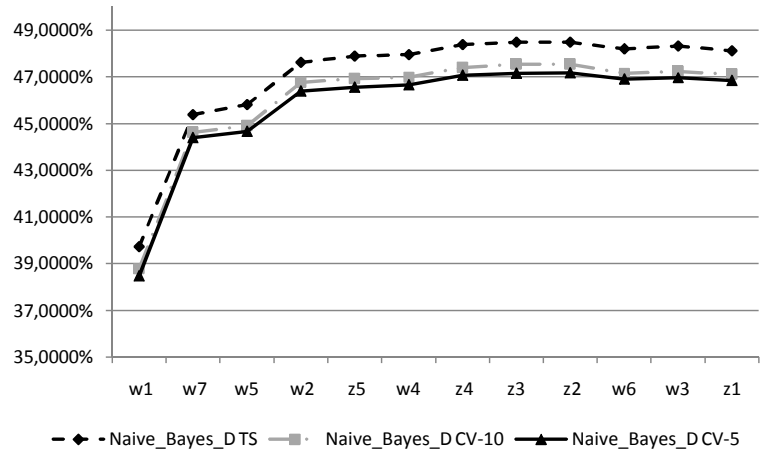

Fig. 2. Prediction accuracy vs used inputs for *Time* dataset

III. EXPERIMENTS WITH CLASSIFIERS

The practical applications of software estimation often use description of software risk or complexity in form of label e.g. high, mid or low. Thus, we decided to use classification as a method of prediction instead of regression that often gives large errors for software. To ensure an unbiased environment, each dataset was filtered to form uniform distribution of its classes. It must be kept in mind, that for 5 uniform classes, the accuracy of blind choice equals to 20%.

Table II contains comparison of accuracy ratios for C4.5, RandomTree (RT in abbrv.) and Classification and Regression Tree (CART in abbrv.) classifiers – the detailed information about these methods may be found in [6], [7], [8] and [9]. The values presented in Table II are the best accuracies achieved by each classifier.

As it can be noticed, the most accurate method for each dataset was RandomTree. This method also did not require careful and sophisticated adjustments of parameters like other methods. It must be mentioned that the most popular decision tree classifier i.e. C4.5 had the largest number of adjusting parameters. That increase searching space in which researchers must be looking for an optimal configuration. Moreover, RandomTree was the fastest algorithm comparing their speed of learning.

IV. EXPERIMENTS WITH METACLASSIFIERS

In the second stage of experiments, we have tested metaclassifiers to check if they were able to increase performance of previously examined classifiers. During this stage of experiments we have used 2 methods of boosting i.e. *AdaBoost* [10], *LogitBoost* [11] and *Bagging* metaclassifier [12], [13].

Adaptive–Boosting (AdaBoot) is a process of iterative classifiers learning, where training sets are resampled according to the weighted classification error. The more errors occur in the class, the bigger weight this class receives. LogitBoost is an another variant of Boosting that uses binomial log-likelihood weight calculating function.

Bagging is an acronym for Bootstrap AGGregatING. Its idea is to create an ensemble of classifiers built using bootstraping of the training dataset. Output of this ensemble is a result of plurality vote. The process of internal classifiers creation might be parallel and therefore their outputs are independent.

Tables III and IV show accuracy of *AdaBoost* and *Bagging* methods with different number of internal iterations. The decision trees described in the previous section were used as the core classifiers in each of metaclassifiers. This part of experiment was performed using *Effects* dataset, because it was as the most difficult of all four datasets.

The results from C4.5 and CART decision trees increased after usage of metaclassifiers. RandomTree, due to its construction and random internal loops, was not affected by *Boosting*. Therefore, we claim that this algorithm is a robust member of decision trees family.

The *Bagging* also caused accuracy increase for all analysed classifiers, but a low number of iterations for this metaclassifier resulted in slightly decrease of it accuracy.

For another boosting method – *LogitBoost* we have compared its accuracy in 2 different variants . This metaclassifier can use resampling or reweighting during boost procedure. Figure 3 presents plots with dataseries for both variants of *LogitBoost*. The core classifier used in *LogitBoost* was *REPTree* being an algorithm of "Fast decision tree learner". It must be noted that value corresponding to 0–iterations is the accuracy for *REPTree* not *LogitBoost*. Thus, an increase of accuracy after *LogitBoosting* can be noticed.

During the evaluation of the results another important characteristic, apart of accuracy ratio, was examined – Receiver Operating Characteristic (ROC) [14]. As it can be noticed on Figure 4, ROC for each class increased with a number of MultiBoost iterations (ROC equals to 1 for an ideal classifier). Similarly to the previous experiments the metaclassifiers were built over *REPTree*.

Figure 5 presents how classification accuracy increases for *MultiBoost*. It can be noticed that this metaclassifier also managed to achieve better prediction results than its core

TABLE II
COMPARISON OF PREDICTION MODELS FOR 4 DATASETS - *Scope*, *Time*, *Costs* AND *Effects*.

Classifier	*Scope*	*Time*	*Cost*	*Effects*
C4.5	97.3560%	67.6481%	78.2700%	55.3124%
RT	99.2803%	71.8745%	91.9815%	76.8332%
CART	96.47%	64.9173%	74.5639%	70.3331%

TABLE III
THE PREDICTION ACCURACY VS. NUMBER OF ITERATION FOR *AdaBoost* MATACLASSIFIER (*Effects* DATASET).

Iterations	*C4.5*	*RT*	*CART*
2	55.3124%	76.8267%	69.1696%
5	62.0127%	76.8267%	75.7681%
10	63.6393%	76.8267%	76.4975%

TABLE IV
THE PREDICTION ACCURACY VS. NUMBER OF ITERATION FOR *Bagging* MATACLASSIFIER (*Effects* DATASET).

Iterations	*C4.5*	*RT*	*CART*
2	58.4172%	72.4245%	53.4018%
5	62.5871%	77.298%	56.6873%
10	64.685%	78.3308%	57.7459%

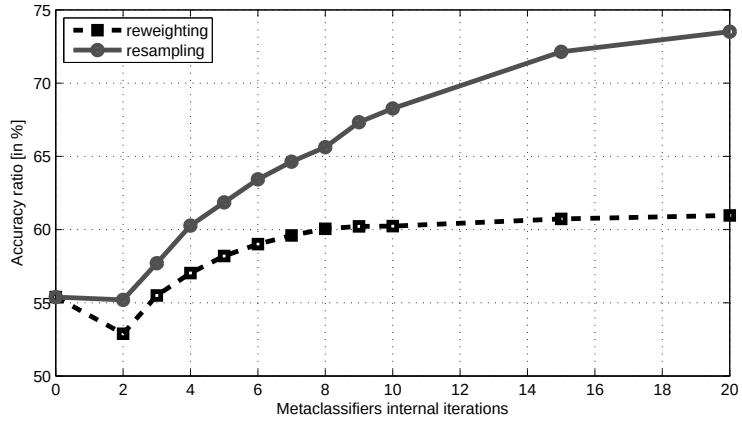

Fig. 3. Classification accuracy for *LogitBoost* (*Effects* dataset)

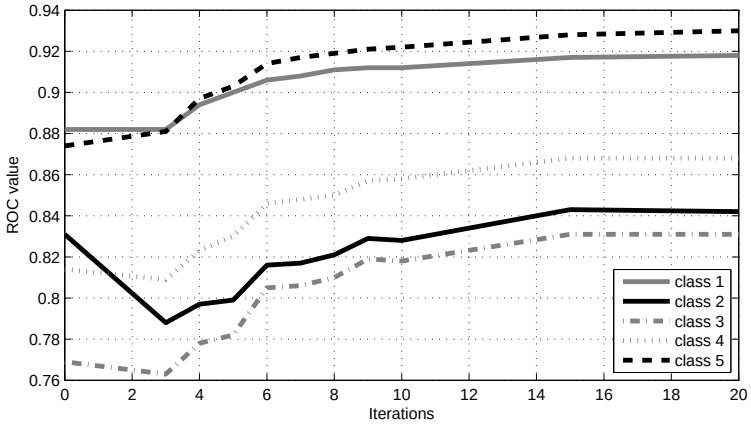

Fig. 4. ROC values for each class vs. no. of iterations for *MultiBoost* (*Effects* dataset)

classifier. Therefore it is possible to claim that metaclassifiers offer a potential to increase estimation accuracy.

V. CONCLUSIONS

The research presented herein prove that Open Source projects management may be supported by data mining methods. As all four datasets presented herein were extracted or calculated

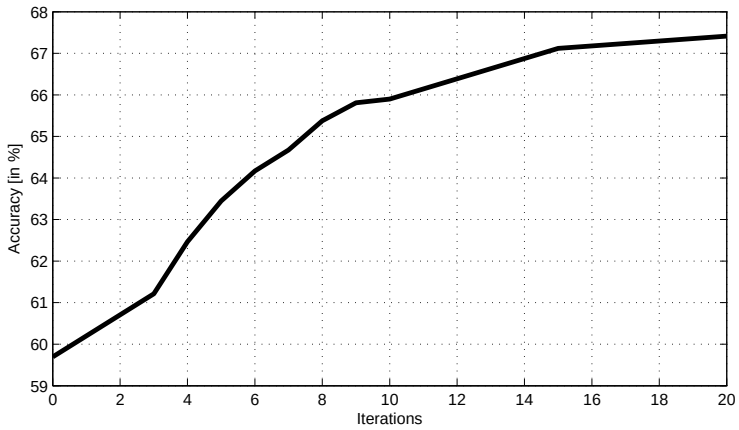

Fig. 5. Classification accuracy vs no. of iterations for *MultiBoost* (*Effects* dataset)

from database fields, designed to store data required by web-based project management platform, it is possible to predict important factors for project without necessity to use any other (external) data sources.

The examined classifiers and metaclassifiers showed large differences in performance e.g. Neural Networks and Support Vector Machines were rejected at early stages, due to their low accuracy for each dataset. It was caused by a large number of unordered labelled attributes. This resulted in decreased performance of Neural Networks and Supporting Vector Machine classifiers, that work well on numeric attributes. However, several methods from decision trees family showed high accuracy for examined data and we claim that it designates them to problems with similar datasets. The next stage of experiments by incorporating metaclassifiers they allowed to increase significantly the prediction accuracy.

It is worth of mentioning, that some methods of classification were practically impossible to use due to their speed or to high memory requirements. In our opinion, researchers focus their attention too much on developing new methods of data processing, that achieve a slight increase of accuracy, forgetting to check if these methods work of fail in front of a complicated tasks.

In the future research, we plan to analyse the dynamics of modelled process, that could lead to better understanding and more effective decision support. We also plan, using created datasets, to analyse computational complexity and requirement of computer resources for popular machine learning algorithms.

ACKNOWLEDGEMENTS

This research was based on SourceForge.net data repository [1].

REFERENCES

[1] G. Madey, "The sourceforge research data archive (srda)," 2008, http://zerlot.cse.nd.edu. [Online]. Available: http://zerlot.cse.nd.edu

[2] U. Raja and M. J. Tretter, "Experiments with a new boosting algorithm," in *Proceedings of the Thirty-first Annual SAS Users Group International Conference*. SAS, 2006.

[3] Y. Gao, Y. Huang, and G. Madey, "Data mining project history in open source software communities," in *North American Association for Computational Social and Organization Sciences*, 2004.

[4] R. English and C. M. Schweik, "Identifying success and abandonment of floss commons: A classification of sourceforge.net projects," *Upgrade: The European Journal for the Informatics Professional*, vol. Vol. VIII, no. 6, 2007.

[5] D. Dzega, "The method of software project risk assessment," Ph.D. dissertation, Szczecin University of Technology, June 2008.

[6] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann Publishers, 1993.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont: Wadsworth International Group, 1984.

[8] L. Breiman, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

[9] G. L. McColm, "An introduction to random trees," *Research on Language & Computation*, vol. 1, pp. 203–227, 2004.

[10] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *In Proceedings of the Thirteenth International Conference on Machine Learning*, 1996, pp. 148–156.

[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, p. 337407, 2000.

[12] L. Breiman, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[13] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken: Wiley-IEEE, 2004.

[14] M. Gonen, *Analyzing Receiver Operating Characteristic Curves Using SAS*. SAS Press, 2007.