Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 361–367
DOI: 10.15439/2014F499
ACSIS, Vol. 2

Building an Ensemble from a Single Naive Bayes Classifier in the Analysis of Key Risk Factors for Polish State Fire Service

Stefan Nikolić, Marko Knežević, Vladimir Ivančević, Ivan Luković
University of Novi Sad, Faculty of Technical Sciences,
Trg Dositeja Obradovića 6, 21 000 Novi Sad, Serbia
Email: {stefan.nikolic, marko.knezevic, dragoman, ivan}@uns.ac.rs

***Abstract*—In this paper, we describe our solution in a competition that required performing data mining to identify key risk factors for the State Fire Service of Poland. The goal was to create an ensemble of Naive Bayes classifiers that could predict incidents involving firefighters, rescuers, children, or civilians. To this end, we first created a single Naive Bayes classifier and then partitioned the set of attributes used in that classifier. The attribute subsets were used to create new Naive Bayes classifiers that would form an ensemble, which generally performs better than both the single classifier and ensemble obtained by searching over all attributes considered when creating the single classifier. The application of our approach yielded a solution that ranked third in the competition.**

I. Introduction

THE main problem in our study is how to use data from incidence reports to identify key factors influencing the risk of serious injuries in actions carried out by the Polish State Fire Service. The dataset and the task description were provided by the organizers of the data mining competition hosted within the framework of the 9th International Symposium on Advances in Artificial Intelligence and Applications (AAIA'14), which is a part of the Federated Conference on Computer Science and Information Systems (FedCSIS) 2014. Additional information about the competition and its propositions is available in .

In accordance with the competition instructions, we performed data mining on the incidence reports. These reports are structured as a single table, in which each row represents one report and each column represents one attribute, i.e., a potential risk factor for injury during interventions and rescue operations. The result of this study is an ensemble of Naive Bayes classifiers that could be used to predict injuries of involved rescuers or civilians based on the most important risk factors.

We describe how we created such an ensemble and compare its performance to that of a single classifier and ensemble obtained by another method that searches over larger set of attributes. The utilized approach represents the main contribution of this paper.

The research presented in this paper was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, Grant III-44010

In addition to Introduction and Conclusion, the paper features four sections. In Section II, we present the dataset and the task in the competition. In Section III, we present our approach to creating both a single Naive Bayes classifier and a Naive Bayes ensemble. In Section IV, we offer the comparison of performance of the single classifier and ensembles formed using the proposed approach. In Section V, we review work related to predicting fires and associated injuries.

II. Problem Description

In this section, we provide an overview of the dataset and outline the shared competition task.

A. Dataset

The provided dataset is extracted from 50,000 reports, which correspond to actions carried out by the Polish State Fire Service within the city of Warsaw and its surroundings between 1992 and 2011. The dataset is highly dimensional and given in form of a table, in which each report is described by 11,852 attributes. All of these attributes are discrete and only a few have more than two possible values. The data are also sparse, since only a small fraction of the attributes has a non-zero value for a particular report. There are three binary decision attributes that describe whether there were casualties among firefighters, children or other involved people, respectively. All three decision attributes are highly imbalanced, since the positive classes correspond to relatively rare events.

B. Task

The task is to identify attributes that could be used to robustly assign reports to corresponding decisions labels. As defined by the organizers, the quality of a solution is assessed by measuring performance of a classifier ensemble composed of Naive Bayes models. Those models are constructed using ten attribute sets, separately for each decision attribute. An output of the ensemble is computed by averaging probabilities of the positive classes returned by

978-83-60810-58-3/$25.00

individual Naive Bayes models. The performance of the ensemble is measured by taking an average Area Under the ROC Curve (AUC) over the probability predictions for each decision attribute, decreased by a penalty for using a large number of conditional attributes. Namely, if we denote the chosen set of attributes by *s*, the total number of attributes used (with repetitions) by |*s*|, and AUC of a classifier ensemble for the i-th decision attribute by $AUC_i(s)$, then the quality measure for the assessment of a chosen set can be expressed as:

$$score(s) = F\left(\frac{1}{3}\sum_{i=1}^{3} AUC_i(s) - penalty(s)\right) \quad (1)$$

$$penalty(s) = \left(\frac{|s| - 30}{1000}\right)^2 \quad (2)$$

$$F(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & otherwise \end{cases} \quad (3)$$

C. Approach to building an ensemble

In this research, we base our work on the task defined in the previous subsection. However, we are also interested in finding a single classifier, using a similar evaluation function, which is calculated as an average of three AUC values decreased by the penalty. Then, we examine whether the single classifier could be divided so to obtain an ensemble yielding higher accuracy compared to the original classifier. We also try some other, more common method for creating an ensemble. Finally, all of these approaches are compared.

III. METHODOLOGY

Our approach to ensemble creation was motivated by the task defined in the previous section. We were first interested in finding a single classifier. Then, we examined how to obtain an ensemble. We used two approaches for creating an ensemble - a division of the single classifier and a search over the same attributes that were considered (searched) when creating the single classifier. In this section, we describe how we select attributes to train a single classifier and how an ensemble may be formed using a custom boosting-based algorithm.

A. Attribute ranking

As already mentioned, the used dataset is highly dimensional. Therefore, in order to find the best classifier (model), it is not practical to search over the entire set of attributes. Hence, we first ranked all attributes according to their relation to the decision attributes. Since the used dataset is large, in order to speed up the process of testing different ranking methods, we manually constructed a small dataset that had similar characteristics as the original dataset, i.e. it included discrete (usually binary) attributes and sparse data. Using this small dataset, we could analyze the results of different ranking methods. The chi-square test [3] gave very good results, so we chose it as our method for attribute ranking.

For each attribute, using the chi-square test, we calculated three weight coefficients, each representing a degree to which an attribute is related to one of the three decision attributes. Finally, for each decision attribute, we ranked all attributes according to their weight.

B. Creating a single classifier

After the ranking, we started a search for a single classifier. Once the search is finished, we should have an approximate accuracy degree that can be reached with one classifier, but also the approximate limit in the number of attributes for a classifier (single or ensemble), since we have a penalty for using a large number of attributes. In order to find the set of attributes that represents the best classifier we used algorithms based on forward and backward search [4].

First, for each of the three decision attributes, we selected the top 50 attributes with respect to their weight rank. The union of these three groups of attributes yielded a set of 121 attributes. We then executed forward search on this set in order to find the best subset, i.e. the one offering the highest accuracy when predicting the values of three decision attributes. The function responsible for evaluating these subsets uses 5-fold cross validation. For each fold, it creates three Naive Bayes models, one for each decision attribute. It calculates predictions using each model and assesses the individual model accuracy using the AUC value. We then calculated the average of these three AUC values and decreased it by the penalty. The overall accuracy is calculated by averaging these values over all five folds.

The forward search process narrowed down the set of 121 attributes to a starting set of 41 attributes, which yield a model whose overall accuracy is 0.950, as calculated by our evaluation function. To improve accuracy, we had to increase the number of attributes being searched. One option was to include more than 50 attributes for each decision attribute and start a forward search from the beginning. However, due to a lack of computing power, we had to gradually increase the number of attributes.

First, for each of the three decision attributes, we selected attributes that were initially ranked between 51 and 75. We created a union of the three sets of attributes with these rankings. This union was used in a forward search that started not from the empty set but from the starting set. This forward search led to the inclusion of new attributes. Next, we performed a backward search, which resulted in the removal of some attributes.

We repeated the whole process and gradually increased the number of attributes to 350. The latest iteration in attribute selection resulted in 72 attributes, which yield a model whose overall accuracy is about 0.965, as calculated by our evaluation function.

The forward search is an iterative process. In each iteration, all attributes that are not already in the model are tested to be included. The attribute that yields the biggest improvement in accuracy is added to the model. Usually, attributes that show improvement in one iteration showed improvement in the previous iteration too. Considering this, in order to speed up the process of searching, we implemented forward search so that in every new iteration it searches only over the attributes that showed improvement in the previous iteration. Also, backward search is implemented in the same manner, so that in every new iteration it only considers attributes whose potential removal showed improvement in accuracy in the previous iteration. In the rest of this paper, whenever we mention the terms forward or backward search, we refer to variants described in this subsection.

It is worth noting that we tried to include additional ranking to speed up the process of searching even more. We tried to select some large number of attributes from each ranking (based on the chi-square test), unite selected attributes and rank them according to the metric that measures how good predictors they are. The metric measures the predictive quality of a single attribute as the accuracy of the model formed using just that one attribute. This additional ranking did not lead to any improvement, i.e. the forward search process did not include attributes with higher weight (according to this metric) as often to justify this method.

C. Creating an ensemble

The main idea behind ensemble systems is to create many classifiers and combine them so that the combination improves upon the performance of a single classifier. Generally, good ensembles should demonstrate diversity, i.e. each classifier should make errors on different examples, so that in combination these classifiers could reduce the total error. Algorithm that we used in order to create a diverse set of classifiers is based on the idea of boosting method, more concretely on the idea behind the AdaBoost algorithm [5].

We want to examine two approaches when creating ensembles. The first one is to try to use one strong single classifier that may be constructed after searching over a set of attributes and to split it into a set of classifiers. The other is to search through that entire set of attributes in order to find a different ensemble. Finally, we may observe differences between two ensembles. In order to make a comparison, these ensembles will have similar structure, i.e. they will include the same number of attributes and the cardinalities of the corresponding classifiers will be the same.

The advantage of the first approach is that it consumes less time, because it searches over a smaller set of attributes. Even if we take into consideration the time needed to find a single classifier, this method is still faster, since our search algorithm needs less time to find one big classifier than ten smaller classifiers where for each of them the search starts from the beginning. We can expect the first method to yield lower accuracy, because it searches over a limited set of attributes. However, it may be less prone to overfitting since it uses only significant attributes selected for a single classifier, but this could be the subject of further analysis.

In this research, we will only consider ensembles in which classifiers have balanced cardinality. Moreover, we will not examine cases when different classifiers in an ensemble may include the same attributes. These two constraints may be part of future research.

D. Boosting-based algorithm for ensemble creation

Our algorithm for ensemble creation follows the general idea of the well-known AdaBoost algorithm. It is an iterative method, where in each iteration we construct one classifier that focuses on examples that are misclassified by previous classifiers.

The algorithm uses ten iterations, as we need ten classifiers. At the beginning of each iteration we construct a training set, by sampling with replacement 50,000 examples from the original training set. Every example has the assigned probability to be sampled. The probability depends on the accuracy with which the example was classified by the previous classifiers (constructed in previous iterations). The more misclassified an example, the higher is its probability to be sampled for the next iteration. Therefore, each new classifier focuses on examples that were misclassified. Initially, the probabilities (p) have uniform distribution, so that in the first iteration every example has the same probability of being sampled.

In each iteration, we search for a classifier that maximizes accuracy, which is calculated as average AUC value for three decision attributes. The selection process is based on the same forward and backward search algorithms that we used when building a single classifier. Once the classifier is found, the probability for each example is updated in accordance with the classification error (err). When classifying, the Naive Bayes method calculates probabilities, so for each example it yields probabilities for three decision attributes. The average of these three probabilities indicates accuracy, and the classification error is calculated as a complement. Classification error is in the interval [0,1]. Here, we introduce a small constant c, which is close to 0, so that probabilities are multiplied with value from interval [c,1]. This constant is used because we do not want the probability of an example classified with error 0 to be multiplied by 0. All the steps executed in one iteration are as follows:

1. Create a training set, using the probabilities p.
2. Using this training set, find the most accurate classifier.
3. Update the probabilities according to the formula: $p = p * (c + (1 - c) * err)$
4. Normalize the probabilities so their sum is 1.

TABLE I
NUMBER OF USED ATTRIBUTES AND RESULTS FOR EACH STEP OF MODEL SELECTION

Number of attributes from rankings	Total number of attributes	Number of attributes in the model	Number of inserted + removed attributes	Overall accuracy of the model
50	121	41	41	0.9500514
75	175	53	14 + 2	0.9561488
100	233	59	6 + 0	0.9571192
150	335	66	9 + 2	0.9586754
200	430	67	7 + 6	0.9605470
250	540	69	6 + 4	0.9630121
300	628	70	2 + 1	0.9638735
350	727	72	4 + 2	0.9648648

If the repetition of attributes among classifiers is not allowed, this algorithm may yield imbalanced classifiers, i.e. classifiers that have very different accuracies. For instance, the first classifier could contain the most significant attributes, because it is created using the uniform distribution of probabilities. Ensembles usually demonstrate better performance if the classifiers are more balanced. Hence, if necessary, we may try to balance these classifiers to improve the overall accuracy.

IV. RESULTS AND DISCUSSION

In this section, we provide an overview of the performance of the single classifier, and ensembles formed by dividing the single classifier and searching over all attributes considered when creating the single classifier. Finally, all of these approaches are compared with respect to the predictive quality of models obtained.

A. Creating a single classifier

As discussed above, in every step of the (model) selection process, we increased the number of attributes being searched. In Table 1, we present the results obtained in each of these steps. Each row contains data about one step in the process. In the first column, we show the number of attributes from each of the three rankings considered in the search process. The second column indicates the total number of attributes considered in the search process, i.e. the union of all attributes obtained from the three rankings. The cardinality and accuracy of the obtained model are presented in the third and fifth column, respectively. Furthermore, the fourth column indicates the number of attributes inserted and removed from the model during one step.

We stopped at the step that included 350 attributes from each of the rankings. Certainly, we could include additional steps in order to improve our model. Each step terminates after a relatively short period. In the future, we can execute further steps, and we expect further improvement in accuracy. We do not expect the cardinality of the model to change very much because of the penalty for using a large number of attributes.

B. Creating an ensemble

Since some of the described methods for creating ensembles are time-consuming, we decided to first test our methods on the initial set used in the process of creating a single classifier (the first row in the Table 1). Later, chosen methods are applied on the final set (the last row in Table 1). The initial set has 121 attributes and yields a classifier that includes 41 of these attributes.

In the first approach, we want to use the classifier and try to split it into ten smaller ones, which are further combined into an ensemble. The cardinalities of these classifiers will be balanced, so there will be nine classifiers with 4 attributes and one with 5.

In order to check the probability that this kind of split yields an ensemble that is more accurate than the single classifier, we made 100 random splits (each split yielding nine subsets with 4 attributes and one with 5). For each of these 100 ensembles, we observed the value that represents accuracy, calculated by the formula defined in the task and using 5-fold cross validation. In the next table (Table 2) we give some quantitative measures regarding these values for all of the 100 ensembles (minimum, maximum, quartiles and mean). The histogram (Fig. 1) shows how the values are distributed.

The accuracy of a single classifier of 41 attributes is 0.9500514. From Table 2 and the accompanying histogram, we may see that there is a large proportion of ensembles whose accuracy is higher than that of the single classifier.

We aim to find a method that yields an ensemble with high accuracy, ideally higher than all of the ensembles obtained after random splits. Here, we test our boosting algorithm to split the classifier (41 attributes). We executed this algorithm with different values of constant c. We present these results in Table 3.

TABLE II
QUANTITATIVE MEASURES FOR ACCURACY OF 100 GENERATED ENSEMBLES

Min.	1st Qu	Median	Mean	3rd Qu	Max.
0.9477	0.9490	0.9496	0.9496	0.9501	0.9515

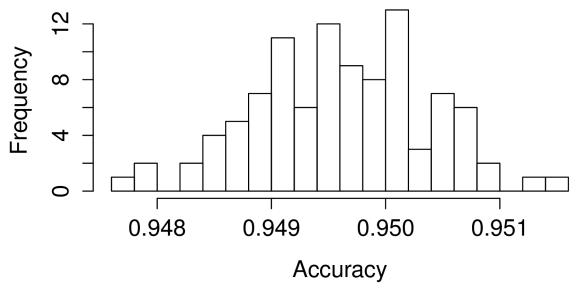

Fig. 1 Accuracy of generated ensembles

TABLE III
ACCURACY OF ENSEMBLES OBTAINED FOR DIFFERENT VALUES OF CONSTANT C

c value	Ensemble accuracy
0.01	0.9494519
0.001	0.9495294
0.0001	0.9498423

As expected, lower values of *c* yield higher accuracy. When we decrease this constant, the examples, with error whose order of magnitude is same as for *c*, have more accurate probabilities. It is possible to decrease the constant further. However, we do not expect this to offer significant improvement, since the number of examples with error lower than 0.0001 is small. We tried to modify our algorithm so that for sampling we use squared (and normalized) probabilities to additionally emphasize differences in probabilities, but this did not produce an ensemble with higher accuracy.

It is important to mention that, when executing our algorithm, we also tried to calculate probabilities according to the original AdaBoost algorithm, but this gave lower accuracy. Most probably, this is because the calculated values used for updating probabilities do not emphasize differences between correctly classified examples and misclassified examples, as much as our algorithm does.

We may see that, when *c* is 0.0001, our algorithm produced an ensemble only slightly more accurate than half of the ensembles obtained by random splits. This does not present a satisfactory result.

One of the reasons why the ensemble produced by the boosting algorithm has low accuracy could be that the classifiers from the ensemble are imbalanced with respect to accuracy. The analysis indicates that the first classifier contains the most significant attributes, because it was created using the uniform distribution of probabilities for sampling. Every subsequent classifier usually has lower accuracy when compared to the previous one.

Among ensembles obtained by random splitting, when we compare those with high accuracy to those with low accuracy, ensembles with higher accuracy usually contain more balanced classifiers. For each ensemble, we may observe the values of accuracy for its ten classifiers and determine standard deviation for these ten values. Lower standard deviation means having more balanced classifiers. In the next Table 4, we show standard deviations for five most accurate and five least accurate ensembles.

From Table 4, it is clear that ensembles with higher accuracy have lower standard deviation. However, this is a characteristic that should be considered, but it is not a rule. When we calculate standard deviation for an ensemble obtained by our boosting algorithm with constant 0.0001, we get a value of 0.1259726. This is much higher deviation when compared to all ensembles from Table 4, and probably from all other ensembles obtained by random splitting. However, the accuracy of this ensemble is better than that of one half of the ensembles obtained by random splitting, as we showed earlier.

Therefore, we will try to balance classifiers in the ensemble obtained by boosting algorithm, with respect to their accuracy. Balancing will be done in iterations. In each iteration, we select classifiers with maximum and minimum accuracy. Then, we try to swap two attributes from those classifiers. We examine each pair of attributes, until the swap yields an improvement greater than some threshold value we set. When the swap is executed, we continue to the next iteration. This process improves accuracy and usually decreases standard deviation. Also, it usually stops after a few iterations. When this method is executed on an ensemble obtained by the boosting algorithm, it stopped after three iterations and improved accuracy to 0.9512469. A small number of iterations is favorable since we do not want to change our ten classifiers too much, considering that boosting algorithms grouped attributes so that each classifier is focused on different set of examples. Accuracy yielded after balancing is high, and only two ensembles from those obtained from random splits have better accuracy.

TABLE IV
STANDARD DEVIATION FOR TEN ENSEMBLES WITH HIGHEST AND LOWEST ACCURACY

ensemble (rank)	standard deviation	ensemble accuracy
1st	0.06989024	0.9515106
2nd	0.07783254	0.9513274
3rd	0.08413265	0.9509978
4th	0.07679123	0.9508524
5th	0.08046396	0.9507565
96th	0.1000637	0.9482818
97th	0.09956434	0.9482052
98th	0.1126992	0.9479637
99th	0.08952922	0.9479142
100th	0.09684023	0.9477181

We also executed balancing on other ensembles that have similar accuracy as ensemble obtained by the boosting algorithm, but none yielded as high improvement. This could indicate that the way in which attributes are grouped by boosting is important, i.e. that the diversity of ensembles is important, and that it may not be easily compensated by balancing. However, we would need to do more testing to confirm that. We also experimented with executing the boosting method on the same set of 41 attributes, where we allowed repetition of attributes among classifiers, but this yielded ten similar classifiers and much lower accuracy.

In our second approach, we executed our boosting algorithm on the set of all 121 attributes. As we aim to compare this approach to the first one, we constructed an ensemble with the same characteristics. That ensemble included classifiers with same cardinalities (nine classifiers with cardinality 4, and one with cardinality 5), with no attribute repetitions. The set of attributes included in this ensemble was much different when compared to the set of 41 attributes from the single classifier. The accuracy of the obtained ensemble is 0.9379888. This accuracy is much lower compared to all ensembles constructed by random splitting.

Finally, considering results presented above, we chose only to test our first approach on the single classifier obtained in the final step. This classifier contains 72 attributes, hence we constructed ensemble containing eight classifiers of seven and two of eight attributes. We tried both boosting with balancing and random splitting. Again, some of the ensembles obtained by random splitting outperformed both the single classifier and the ensemble obtained by boosting and balancing. Our final result was an ensemble yielding the highest accuracy.

V. Related Work

There are many attempts to use fire related data for predicting, managing and reducing injuries caused by fire outbreaks. One group of such studies estimates future wildfire activities in order to reduce negative effects and facilitate its management, limiting the most destructive aspects of fire. Beckage and Platt [6] explored potential of a time series model that considers the area burned in previous years and the Southern Oscillation Index (SOI) condition to predict the area burned in the current years. However, the authors raise attention that their prediction is not true for "out of sample" predictions since model selection, parameter estimation, and the modeling process itself used the data that were to be predicted. Furthermore, Yue, et al. [7] managed to improve their regression and parameterization models for predicting wildfire activity by combining them with an ensemble of climate model projections for 2050 conditions. Moreover, they observed importance of considering meteorological attributes other than temperature in predicting changes in area burned.

The second group of research focuses on predicting injuries and identifying key factors influencing the risk of injuries in various incidents. One such analysis of data from population-based case-control study in King County, Washington [8] showed that there is association between smoking and residential fire injuries. By using relative risk estimation as a uniform method of comparison Warda, et al. [9] summarized house fire injury risk factor data retrieved from fifteen relevant articles. The authors stress that the research is relevant for developing, targeting, and evaluating preventive strategies. In their study, Burgess, et al. [10] compared injury rates among various fire departments and observed that the rates vary substantially. Variations in work practice and risk management regulations as well as in reporting practices were proposed as a potential explanation for the study findings.

Selection of significant attributes is deemed a prerequisite for constructing an accurate prediction model. In this sense, Shai [11] examined social and demographic predictive factors using multiple regression. Besides identifying significant attributes in predicting fire injuries for the civilian population, the author observed interaction effect between age of housing and income. The results of such research showed that older housing, low income, the prevalence of vacant houses, and ability to speak English (native language for area of Philadelphia) have significant effects on fire injury rates. In this research, we also aimed to find features that make accurate predictions regarding injuries in fire incidents. As opposed to Shai, we used Naive Bayes (NB) models, feature ranking and feature subset selection methods. The combination of multiple feature selection methods allowed us to capture complex feature interactions and select the model with the highest predictive capacity. Moreover, we used these methods to create both a single classifier and classifier ensemble, and to compare the two approaches. It was shown that ensemble models can yield higher accuracy than single models [12].

VI. Conclusion

In this paper, we present an approach for creating an ensemble of Naive Bayes classifiers from a single classifier. The creation of this approach was motivated by the need to identify key risk factors for the Polish State Fire Service as part of a data mining competition. The use of the proposed approach yielded an ensemble that generally performs better than both a single classifier and ensemble obtained by searching over all attributes that were considered when creating the single classifier. Moreover, the trained ensemble ranked third in the competition.

The advantage of this approach is that the model can be incrementally improved relatively quickly. We can include more steps in the process of selection of a single classifier in order to improve the classifier. Every step requires relatively small amount of time to finish. After executing these steps, we can make a division of the single classifier in order to obtain a new ensemble. The process of division does not consume much time.

In the future research, we would like to improve our methodology. As our results suggest, one single classifier can be divided into smaller ones which in combination yield higher accuracy. Our goal is to find a method for classifier division that would give a good ensemble, i.e., an ensemble that would outperform all of the ensembles generated by random divisions. Our method did not give completely satisfactory solution, but it is on the line of achieving the goal. So far, we tested only an algorithm based on boosting. It was used for division and when constructing an ensemble by searching (all attributes considered when creating the single classifier). We could try to improve this algorithm, but also to try other different algorithms for creating ensembles, since there are many others known in literature. Later, we should make further comparisons between approaches where we construct an ensemble from a single classifier and by searching. Finally, we could experiment with the possibility of attribute repetitions and different cardinalities among classifiers in an ensemble in order to obtain higher accuracy.

REFERENCES

[1] (2014, 02. June). *AAIA'14 Data Mining Competition*. Available: https://fedcsis.org/2014/dm_competition

[2] (2014, 02. June). *AAIA'14 Data Mining Competition: Key risk factors for Polish State Fire Service*. Available: http://challenge.mimuw.edu.pl/contest/view.php?id=83

[3] P. Romanski, "FSelector: Selecting Attributes," *Vienna: R Foundation for Statistical Computing,* 2009.

[4] E. Cantu-Paz, S. Newsam, and C. Kamath, "Feature Selection in Scientific Applications," in *Proc. ACM International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 788-793, http://dx.doi.org/10.1145/1014052.1016915

[5] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, 1995, pp. 23-37, http://dx.doi.org/10.1007/3-540-59119-2

[6] B. Beckage and W. J. Platt, "Predicting severe wildfire years in the Florida Everglades," *Frontiers in Ecology and the Environment,* vol. 1, pp. 235-239, 2003, http://dx.doi.org/10.1890/1540-9295(2003)001[0235:PSWYIT]2.0.CO;2

[7] X. Yue, L. J. Mickley, J. A. Logan, and J. O. Kaplan, "Ensemble projections of wildfire activity and carbonaceous aerosol concentrations over the western United States in the mid-21st century," *Atmospheric Environment,* vol. 77, pp. 767-780, 2013, http://dx.doi.org/10.1016/j.atmosenv.2013.06.003

[8] J. E. Ballard, T. D. Koepsell, and F. Rivara, "Association of smoking and alcohol drinking with residential fire injuries," *American journal of epidemiology,* vol. 135, pp. 26-34, 1992.

[9] L. Warda, M. Tenenbein, and M. E. Moffatt, "House fire injury prevention update. Part I. A review of risk factors for fatal and non-fatal house fire injury," *Injury Prevention,* vol. 5, pp. 145-150, 1999, http://dx.doi.org/10.1136/ip.5.2.145

[10] J. L. Burgess, M. Duncan, J. Mallett, B. LaFleur, S. Littau, and K. Shiwaku, "International comparison of fire department injuries," *Fire Technology,* pp. 1-17, 2013, http://dx.doi.org/10.1007/s10694-013-0340-y

[11] D. Shai, "Income, housing, and fire injuries: a census tract analysis," *Public health reports,* vol. 121, 2006.

[12] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information fusion,* vol. 6, pp. 83-98, 2005, http://dx.doi.org/10.1016/j.inffus.2004.04.003