Analysis of Ensemble Methods for Security Classification in Multi-Level Security Systems

Paal E. Engelstad

Abstract—Research that explores the use of machine learning for automatic security classification of information objects is about to emerge. In this paper, we investigate the use of ensemble methods to this problem. To the best of our knowledge, this has not been addressed in previous works. We compare various ensemble methods with traditional methods that has been investigated in existing works. We also discuss and analyze to which extent the additional computational costs of the ensemble methods are worth the potential performance benefits.

Index Terms—Multi-level security, classification, machine learning, ensemble methods, feature selection, cross-domain information exchange.

I. INTRODUCTION

Security classification is about classifying information objects, such as documents and text messages, into different groups, e.g. "Secret", "Confidential", "Unclassified" etc. The concept is not only used by the military, government agencies and international organizations, but also by private corporations, e.g. see [1]. The classification indicates the sensitivity of the contents of different information objects and mandates how the information object shall be treated according to the governing security policy.

The content of an information object is typically classified by human inspection and assessment, and given a security label (e.g. "Public" vs "Confidential" or "Business Internal"). However, with an increasing amount of generated information, there is a need for tools that can assist with automatic security classification. As a result, research that explores the use of machine learning for automatic security classification of information objects is about to emerge.

Previous works that have applied machine learning to this problem, such as [2], [3], [4] and [5], have focused only on Support Vector Machines (SVM), k-Nearest Neighbor (kNN), Naïve Bayes (NB) and Least Absolute Shrinkage and Selection Operator (Lasso) [6] [7] [8]. To the best of our knowledge, the application of ensemble methods, on the contrary, has not yet been studied for this problem. An in-depth analysis of this is the main contribution of this paper.

II. BACKGROUND

A. Overview of ensemble methods

An overview of ensemble learning is found in [9], which can be used as a starting point for further details and references. Due to the space limitations of this paper we will only give an outline:

The Bayes Optimal Classifier is an ensemble of all the hypotheses in the hypothesis space, where each hypothesis is given a vote that is proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true. It can be shown that on average, no other ensemble can outperform it. However, for most realistic problems the Bayes Optimal Classifier cannot be practically implemented.

Bagging (an abbreviation for Bootstrap aggregating), on the other hand, is easier to implement, since it lets each model in the ensemble vote with equal weight. Usually, each model in the ensemble is trained with a randomly drawn subset of the training set, in order to increase the model variance. In fact, the random forest algorithm combines random decision trees with bagging to increase the classification accuracy.

Boosting, on the other hand, builds the ensemble step-by-step. When adding a new model instance, the training of this instance is focusing on the observations that the existing model instances in the ensemble are mis-classifying. On the one hand, boosting can yield better accuracy than bagging, but on the other hand, it also tends to be more likely to over-fit the training data. By far, the most popular version of boosting is Adaboosting.

A "Bucket of models" can also be considered as an ensemble. However, here a model selection algorithm is used to choose the best model for each problem. There is usually an initial "bake-off contest" between different models, where the best model is selected through cross-validation on the training set. The idea is to test out all models on the training set, and select the best one here for the job of doing the classification of the test set.

In addition comes a range of other ensemble methods, such as Stacking (i.e., training a learning algorithm to combine the predictions of several other learning algorithms), Bayesian Model-Averaging (BMA) and Bayesian Model Combination (BMC). Going into the details of these is outside the scope of this paper.
TABLE I: Popular ensemble method packages available in the statistical package R. Further references for the various methods and algorithms are found in [10]

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Package</th>
<th>Lead author</th>
<th>Algorithms</th>
<th>Release year</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipredBagging</td>
<td>ipred</td>
<td>Peters</td>
<td>Bagging</td>
<td>2002</td>
</tr>
<tr>
<td>RForest</td>
<td>randomForest</td>
<td>Brennan</td>
<td>Random Forest</td>
<td>2002</td>
</tr>
<tr>
<td>LogitBoosting</td>
<td>eaTools</td>
<td>Tuszynski</td>
<td>logitBoost</td>
<td>2005</td>
</tr>
<tr>
<td>AdaBagging</td>
<td>ada</td>
<td>Alfaro</td>
<td>AdaBoost+Bagging</td>
<td>2006</td>
</tr>
<tr>
<td>AdaBoosting</td>
<td>ada</td>
<td>Culp</td>
<td>AdaBoost+Friedman</td>
<td>2006</td>
</tr>
</tbody>
</table>

B. Ensemble method implementations in R

The reader might refer to [10] for an overview of various popular ensemble method packages that are available in the statistical package R. Some of the most popular ones are outlined in Table I.

Even though the algorithms implemented in older packages (e.g. bagging in ipred) has been incorporated into other newer evolved methods (e.g. into RandomForest), the older packages might remain with unique features (e.g. ipred has the ability to include more than one type of base learner).

Due to the large amount of available ensemble methods, it is not possible to cover each and every method here. Thus, we will limit our analysis to the methods listed in Table I. We argue that the methods listed in the table are quite representative for the various ensemble methods in popular use. Indeed, bagging and boosting in general, and AdaBoosting in particular, are heavily used ensemble methods today. Furthermore, the RandomForest algorithm often ends up amongst the best contenders in various machine learning competitions.

III. MAIN EXPERIMENT

A. Experimental setup

The analysis presented in this paper is based on a benchmark experiment for binary classification described in detail in [3], [4] and [5]. Due to space limitations, the reader is referred to these references for an extensive description. The setup will be only briefly summarized in the following.

In our experiments we used a corpus of documents retrieved from the Digital National Security Archive - DNSA [11]. We used around 2800 documents in our analyses, where around half of the documents are "Classified", while the other part is "Unclassified". The document texts are treated as bag-of-words, where we apply word stemming to each word. Each document gives a term-vector with a dimensionality corresponding to the number of available words, and with each vector component providing the frequency of the word used, i.e. it is typically a sparse vector. Vectors of all documents comprise the Document-Term Matrix (DTM), where each row in the matrix is a word-frequency vector corresponding to one specific document. Some specific keyword terms were ignored. We ignore infrequently used words (used less than 15 times), reducing the numbers of columns in the DTM from around 23000 word stems, down to around 5840 term stems. Finally, we used "term frequency-inverse document frequency" (tfIdt) for term weighting [3] [4].

For the machine learning we applied the ensemble methods listed in Table I. An important experimental parameter of the ensemble methods is the size of the ensemble (often referred to as either the number of trees, the maximum number of iterations, ntrees, nbagg, mfinal, etc. ). To start off with a comparable parameter setting, we set the ensemble size to 25, which is the default value for the bagging in the original ipred package in R. (Later in the paper, we will also analyze different other settings of this parameter, as it might affect both performance and computation time.)

In addition to these ensemble methods, we also use Lasso and SVM, since these two methods have shown the best performance of the algorithms presented in previous works [2], [3], [4] and [5]. Lasso and SVM serves as benchmark methods representative of state-of-the art non-ensemble methods.

In the experiments presented in this paper we use a 10-fold cross-validation to decrease the variance in our results. This represents a difference from the analysis presented in [3], [4] and [5], where a 10-fold cross-validation on the training set was used only for tuning machine learner parameters.

IV. RESULTS

A. Classification accuracy

Table II shows the main results derived according to the experimental setup described above. We observe in the left column (labeled with "Acc") that Lasso, SVM, AdaBoosting and RandomForest yield the highest classification accuracy, and that there are no significant differences in the results between these algorithms. AdaBagging is performing somewhat worse, while LogitBoosting is lagging seriously behind.

B. ROC, AUC and False Positives

Previous works have noted that automatic security classification will not be used without human control, intervention and assistance [3] [4]. The reason is that automatic security classification is different from the vast number of machine learning scenarios in the way that mis-classifications can have extremely damaging consequences. If an information object that contains classified information is mistakenly predicted to be unclassified, it might lead to information leakage. This can be especially critical in military settings.

A consequence of this is that only looking at the Classification Accuracy alone might not be sufficient. A further refinement is to study the importance of False Negatives.

TABLE II: Main results

<table>
<thead>
<tr>
<th>Machine learner</th>
<th>Randomized in time</th>
<th>Acc</th>
<th>AUC</th>
<th>FPs</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSO</td>
<td>.783</td>
<td>.869</td>
<td>.070</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>SVM</td>
<td>.792</td>
<td>.862</td>
<td>.064</td>
<td></td>
<td>913</td>
</tr>
<tr>
<td>ipredBagging</td>
<td>.801</td>
<td>.872</td>
<td>.067</td>
<td></td>
<td>6740</td>
</tr>
<tr>
<td>LogitBoosting</td>
<td>.740</td>
<td>.789</td>
<td>.086</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>AdaBagging</td>
<td>.777</td>
<td>.848</td>
<td>.052</td>
<td></td>
<td>11247</td>
</tr>
<tr>
<td>AdaBoosting</td>
<td>.808</td>
<td>.878</td>
<td>.082</td>
<td></td>
<td>13293</td>
</tr>
<tr>
<td>RandomForest</td>
<td>.798</td>
<td>.871</td>
<td>.048</td>
<td></td>
<td>58564</td>
</tr>
</tbody>
</table>
In this paper we define "CLASSIFIED" as negative (N) and "UNCLASSIFIED" as positive (P). This definition has no impact on the relevance of the ROC curves. The reason is that the False (F) and True (T) Positive (P) and Negative (N) rates (R) are interrelated as \( TPR = (1-FNR) \) and \( TNR = (1-FPR) \). This means that if choosing opposite definition, the ROC curve will only be inverted (i.e. mirrored across the \( y=1-x \) line in the graph), and the Area Under the Curve (AUC) will be the same.

To the best of our knowledge, all previously published works on automatic security classification so far have only studied the classification accuracy (Acc) and overlooked the importance of the ROC. The ROC curves for the machine learners are shown in Figure 1.

Usually, the more up to the left the ROC curve is, the better, and therefore the Area Under the Curve (AUC) is a commonly used measure for how "good" a ROC curve is. The AUCs that we obtained for the different machine learners are given in Table II. It shows that the AUC performance of Lasso is comparably the highest relative to its classification accuracy, while LogitBoosting performs even worse than the accuracy indicates. The AUC performances of the remaining machine learners are pretty much consistent with what their accuracies show.

We also conducted a Precision-Recall (PR) analysis, which is important to double-check the results of the ROC and AUC analysis [CITE]. Due to space limitations, we are not going into details here. The PR curves were much in line with the results of the ROC curves (perhaps due to the fact that the corpus is not very skewed).

C. False Positive share (FPs)

Since we are especially concerned with the False Positives (FP) through the Receiver Operating Characteristic (ROC).

The results show that LASSO is extremely quick compared to the other alternatives. LogitBoosting and SVM are comparably slow, but much quicker than the remaining ensemble methods.

B. Why is computation time essential?

In many of the scenarios that are targeted by Automatic Security Classification, there will be an ongoing production of new information objects that need to be labeled with its classification level. A machine learner is using the existing information objects as a training set, in order to predict the classification level of the new objects.

In a previous work [5], it has been shown that there can be an evolution of the topics that are described in the documents, as an "aging" of the information. The most recent of the existing documents are the ones in the training set that are the best predictors of the classifications of new objects, while the oldest documents in the training set might describe outdated topics or have outdated perspectives. It is important to get newly produced information objects into the training set, and train the machine learner regularly so that the machine learner always is trained with the most pertinent information. The computation time of the machine learner determines how quickly it is possible to update the machine learner with newly acquired information.

In such scenarios, all the documents in the training set are older than the documents that shall obtain a predicted security classification. To provide a pertinent analysis of the performance, one must create a similar experiment; i.e., all the documents in the training set must be older than all the documents in the test set. Nevertheless, many analyses, including the analysis we have presented above, uses a cross-validation technique where different folds are determined by randomly selecting documents to the training set and test set. This means that the information objects are not chronologically ordered in time in terms of the machine learning. This yields artificially good results, because the machine learner is not suffering from
seen by the red line, which shows how the moving average and the classification accuracy is decreasing. It is most easily be classified, the training set gets more and more outdated, is quite good. However, as more and more documents are to classification accuracy of the first documents to be classified machine learner is illustrated in Fig. 2. We observe that the consequences of not updating the information aging, and how important it is to frequently update this performance drop by introduction of time interaction techniques [5], but this issue is out of scope here.)

<table>
<thead>
<tr>
<th>Machine learner</th>
<th>Chronological</th>
<th>Accuracy</th>
<th>95% conf.int.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>No</td>
<td>0.77</td>
<td>(0.74, 0.81)</td>
</tr>
<tr>
<td>SVM</td>
<td>Yes</td>
<td>0.70</td>
<td>(0.67, 0.73)</td>
</tr>
<tr>
<td>Lasso</td>
<td>No</td>
<td>0.78</td>
<td>(0.75, 0.82)</td>
</tr>
<tr>
<td>Lasso</td>
<td>Yes</td>
<td>0.74</td>
<td>(0.71, 0.77)</td>
</tr>
</tbody>
</table>

The aging of information between the training set and the test set.

We conducted an additional analysis where a 10-fold cross-validation is conducted on a training set (70% of the corpus) to tune the parameters, before the performance of the machine learner is determined on the test set (Table III). To demonstrate our point, it is sufficient to limit ourselves to Lasso and SVM, which both are quick machine learners that are easily tuned. Indeed, results in Table III show a considerable drop in performance with chronological ordering, i.e., when we ensure that all documents in the training set are older than all documents in the test set. (In fact, it is possible to reduce this performance drop by introduction of time interaction techniques [5], but this issue is out of scope here.)

This performance drop underlines the significance of the information aging, and how important it is to frequently update the machine learner. The consequences of not updating the machine learner is illustrated in Fig. 2. We observe that the classification accuracy of the first documents to be classified is quite good. However, as more and more documents are to be classified, the training set gets more and more outdated, and the classification accuracy is decreasing. It is most easily seen by the red line, which shows how the moving average is decreasing. However, after a certain time, the red line stabilizes. This indicates that the training set consists of both a time-independent basis and a component that is highly time relevant. To always be operating in the left side of the figure where the classification accuracy is at its highest, one needs to update the machine learner sufficiently often.

In scenarios where the information aging is relatively high, the quickest machine learners in Table II, such as LASSO, might perform better than the slowest ensemble methods even though the latter displays good accuracy, since their training gets more outdated than the training of the quickest methods.

It is possible to obtain these performance gains by artificially speeding up the machine learning by reducing the size to the DTM. A feature selection (FS) method can be used to reduce the number of columns of the DTM by ignoring the least significant word stems. However, too aggressive features selection might punish the performance of the machine learner.

In summary, there is a trade-off between the performance gain of the FS (through higher speed and thus having an up-to-date test set) and the performance loss of the FS (through lower overall precision of the information extracted from the training set). This trade-off will be analyzed in further depth in the following.

### VI. Feature Selection (FS)

#### A. Different FS methods

Information Gain (IG), Chi-square analysis (CHI) and Document Frequency (DF) represent some of the different approaches for determining which features to select and which to ignore [12]. We will base our analysis on IG in the following, since previous works on automatic security classification have shown that IG is a good candidate for FS [2] [4].

#### B. Performance losses of FS

Fig. 3 shows how the classification accuracy deteriorates as the feature selection gets more and more aggressive (i.e., as we move more and more to the left on the curves in the figure). Note that the x-axis is logarithmic; the figure shows that one might reduce the size of the DTM quite significantly (down to around 3% of its original size) without losing too much performance. As we reach the black dashed vertical line (which is situated at $x = 10^{-1.5} = 0.0316$ the classification accuracy begins to take a toll. We also observe that the different machine learners are punished comparably equally.

For the AUC, on the contrary, we observe in Fig. 4 that AdaBoosting, LASSO and SVM preserve their AUC performance quite well as the FS gets aggressive, while the other machine learners suffer more.

#### C. Performance gains of FS

Fig. 4 shows that the calculation speeds of almost all the machine learners are directly proportional with the size of the DTM (i.e., here we use a linear scale on the x-axis). The only exception is that RandomForest displays an exponential performance. For relatively small DTMs it is very quick, however, as the DTM gets big the computational overhead increases.
might grow to an insurmountable value. At a DTM with only around 5800 columns, RandomForest took more than 16 hours to complete, while LASSO only spent 7 seconds. In comparison, the difference in classification accuracy between the two is not very big (Table II). In fact, we started out with a DTM of around 23000 columns. In is not unforeseeable that in a realistic scenario the DTM will be much bigger than this.

D. Tuning of the ensemble sizes

Tuning of parameters is quick for LASSO and SVM, while for the slow ensemble methods it is a time consuming process. Nevertheless, we are interested in seeing how the ensemble size affects performance. Due to the time consumption of tuning a full-scale DTM, we conduct a tuning after having carried out a feature selection down to a share of \( x = 10^{-1.5} = 0.0316 \) (vertical black dashed line in Fig. 3).

In Fig. 6 and Fig. 7 we see that the performance suffers when selecting an ensemble size of less than 20 (while we chose the default ipred value in R of 25). After this point, the performance gain of increasing the ensemble size decreases more and more.

As for the computational overhead, on the contrary, the speeds of all the ensemble methods studied are - as expected - linearly dependent on the size of the ensemble (Fig. 8). This means that the ensemble sizes will be kept quite low in many realistic scenarios, due to the aforementioned trade-off.

Our benchmark methods, LASSO and SVM, are not dependent on an ensemble. For comparison, their performance is shown as horizontal lines in Figs. 6, 7 and 8.
AdaBoosting LASSO RF

RandomForest LASSO RandomForest

In general and preserves both the accuracy and the AUC very
displays good classification accuracy and AUC performance
semble methods turned out to be promising. AdaBoosting
overhead of the training.
taking into account the ROC, PR, AUC and the computational
area, by not only considering classification accuracy, but also
deepen analysis than previously presented within this research
for this problem in previous works. In addition, we provide a
To the best of our knowledge, this has not been considered
for Lasso and SVM are flat, and for clarity drawn deliberately
only in the right part of the figure.

One of the main contributions of this paper is to analyze the
use of ensemble methods for automatic security classification.
To the best of our knowledge, this has not been considered
for this problem in previous works. In addition, we provide a
deeper analysis than previously presented within this research
area, by not only considering classification accuracy, but also
taking into account the ROC, PR, AUC and the computational
overhead of the training.

Looking at only the classification performance, the en-
ssemble methods turned out to be promising. AdaBoosting
displays good classification accuracy and AUC performance
in general and preserves both the accuracy and the AUC very
well with aggressive Feature Selection (FS). AdaBagging and
RandomForest perform a little worse in terms of accuracy and
AUC, but yields a low share of False Positives, which is a
main concern in these scenarios.

We used LASSO and SVM as benchmark methods. LASSO
turns out as a compelling alternative to ensemble methods.
Like for AdaBoosting, the classification accuracy and AUC
are comparably good and well preserved with aggressive FS.

More importantly, LASSO is extremely quick to train,
compared to the ensemble methods. The paper brings to
attention the importance of the computation overhead for many
scenarios targeted by automatic security classification, and that
there is a trade-off between speed and the other performance
metrics (such as Accuracy and AUC) of the machine learners.
Due to an exponential increase in computation time with
increasing DTM size, RandomForest seems to suffer a lot,
and might not be appropriate for many automatic security
classification scenarios.

For the other ensemble methods, on contrary, the compu-
tation speeds are linearly dependent on the size of both the
ensemble and the DTM. The classification accuracy and ACU,
on the other hand, were preserved well when we decreased
ensemble and DTM sizes from relatively common/medium-
sized values. Then, it might make sense to reduce the ensemble
size and DTM sufficiently, due to the aforementioned trade-off
that involves the computation speed. However, studying this
trade-off in further detail is left for future work.

VII. CONCLUSIONS AND FUTURE WORK

One of the main contributions of this paper is to analyze the
use of ensemble methods for automatic security classification.
To the best of our knowledge, this has not been considered
for this problem in previous works. In addition, we provide a
deeper analysis than previously presented within this research
area, by not only considering classification accuracy, but also
taking into account the ROC, PR, AUC and the computational
overhead of the training.

Looking at only the classification performance, the en-
ssemble methods turned out to be promising. AdaBoosting
displays good classification accuracy and AUC performance
in general and preserves both the accuracy and the AUC very
well with aggressive Feature Selection (FS). AdaBagging and
RandomForest perform a little worse in terms of accuracy and
AUC, but yields a low share of False Positives, which is a
main concern in these scenarios.

We used LASSO and SVM as benchmark methods. LASSO
turns out as a compelling alternative to ensemble methods.
Like for AdaBoosting, the classification accuracy and AUC
are comparably good and well preserved with aggressive FS.

More importantly, LASSO is extremely quick to train,
compared to the ensemble methods. The paper brings to
attention the importance of the computation overhead for many
scenarios targeted by automatic security classification, and that
there is a trade-off between speed and the other performance
metrics (such as Accuracy and AUC) of the machine learners.
Due to an exponential increase in computation time with
increasing DTM size, RandomForest seems to suffer a lot,
and might not be appropriate for many automatic security
classification scenarios.

For the other ensemble methods, on contrary, the compu-
tation speeds are linearly dependent on the size of both the
ensemble and the DTM. The classification accuracy and ACU,
on the other hand, were preserved well when we decreased
ensemble and DTM sizes from relatively common/medium-
sized values. Then, it might make sense to reduce the ensemble
size and DTM sufficiently, due to the aforementioned trade-off
that involves the computation speed. However, studying this
trade-off in further detail is left for future work.

REFERENCES

[1] W. Nicolls, “Implementing company classification policy with the
Proceedings of The 16th International Workshop on Information Security
Applications (WISA 2015 ), Jeju Island, Korea, August 20-22, 2015.
classification lists (dirty word lists) for automatic security classification,”
Proceedings of The 7th IEEE International Conference on Cyber-
enabled distributed computing and knowledge discovery (CyberC, 2015),
Cyber Security and Privacy (CSP), Xian, China, Sept 17-19, 2015.
Proceedings of The 7th IEEE International Conference on Cyber-
enabled distributed computing and knowledge discovery (CyberC, 2015),
Cyber Security and Privacy (CSP), Xian, China, Sept 17-19, 2015.
classification methods based on the type of attributes and sample size,”
for generalized linear models via coordinate descent,” Journal of
http://www.jstatsoft.org/v33/i01/
in text categorization,” Proceedings of ICML-97, 14th International