BGP Anomaly Prediction Using Ensemble Learning

Marijana Cosovic and Emina Junuz

***Abstract*—This paper investigates anomalies such as worms, power outages, and routing table leak (RTL) events occurring in Border Gateway Protocol (BGP) that can cause connectivity and data loss issues. Ensemble learning is a machine learning model employing multiple classifiers in order to reliably identify network anomalies. We use bagging, boosting, and random forests ensemble models trained on network anomaly datasets for classification improvement. Models were compared with respect to the following performance metrics: F-measure, Matthews correlation coefficient (MCC), Receiver operating characteristic (ROC) curve, precision-recall (PR) curves and model execution time. We observed improvement in performance measures when ensemble classifiers realized in Python were used in comparison to our previously reported results on single classifiers. Further improvement in most performance measures was observed by using sampling techniques (oversampling and undersampling) on anomalous datasets. This approach increases model execution time which is not favorable for real-time anomaly detection models.**

***Index Terms*—BGP, bagging, boosting, random forest.**

I. Introduction

The main purpose of the Border Gateway Protocol (BGP) is to exchange routing and reachability information between autonomous systems (AS) in the Internet. BGP is susceptible to anomalous behavior (intentional or unintentional), and as such can cause problems with connectivity and data loss. BGP anomaly detection is an ongoing challenge for researchers in the computer networking domain as well as network operators. Machine learning techniques have been used to address anomaly prediction within the Internet [1]-[4].

The standard for classifier evaluation are various performance measures. By combining multiple classifiers in ensemble methods we aim to improve performance measures previously reported in our work [1]-[3]. Authors is [4] developed an unsupervised ensemble anomaly detection method for network traffic. Although, it has no need for prior data labeling, considering it uses multiple baseline models trained on multiple sets od sampled data, it provides no improvement in performance measures in respect to supervised methods.

We have formerly employed Support Vector Machines (SVM), Naïve Bayes (NB), and decision tree (C4.5) classifiers, as well as neural networks [1]-[3] for anomaly detection. In this paper we use ensemble methods, namely bagging [5], boosting [6] and random forest [7], to achieve better classification results. We do not consider accuracy as one of the performance measures since anomalous datasets are highly imbalanced. Instead we use F-measure, Matthews correlation coefficient (MCC), Receiver operating characteristic (ROC) curve, and precision-recall (PR) curves. In addition, we consider model execution time, since anomaly detection systems aim at real-time performance. We observe BGP routing data and traffic traces contained in the BGP update messages and extract information during anomaly occurrence. We train classifiers based on information extracted from the BGP update messages for ten different anomalous events: worms, power outage event, and routing table leak events affecting BGP performance.

This paper is organized as follows. In Section II, we describe methodology used for BGP anomaly classification. Three ensemble models are briefly presented and their performance measures obtained on all ten anomalous datasets are discussed in Section III. In addition, oversampling and undersampling algorithms were applied to original datasets and performance measures achieved on those datasets were discussed. We conclude with Section IV.

II. Methodology

Classification of BGP anomalies was performed on ten different datasets of known anomalies in BGP. Slammer, Nimda, and Code Red I are worm events considered in [1]. AS9121, AWS, Malay, and Indosat are RTL events described in detail in [2] while Moscow Power Blackout, Panix Hijack, and AS-PATH Error events were discussed in [3]. Web application [8] was developed for analysis and testing of the anomalous events in the Internet. For BGP anomaly events considered, data is collected through The Réseaux IP Européens (RIPE) [9] and Route Views [10] projects. Data preprocessing techniques have been used to complete and clear inconsistencies in the datasets. We did not consider that a small percentage of missing data (less than 0.5%) could have a negative impact on classification, but nevertheless missing data is replaced by the average of ten surrounding values in the neighborhood of the missing value. Since feature matrices are generated for a period of five days, a large number of files corresponding to one event are integrated into one. We extracted 15 features from the BGP update messages of anomalous events [8], considering volume and AS-PATH features, hence our feature matrix for each of the datasets is 7200x15. Data is normalized (each feature has a mean value of zero, and standard deviation of one) to eliminate the impact of Internet growth (number of ASs, routing table, etc.), since

Manuscript received February 11, 2019; revised June 2, 2019. This work was supported in part by the Ministry of Science and Technology, Republic of Srpska, Bosna and Hercegovina.

M. Cosovic is with the University of East Sarajevo, Faculty of Electrical Engineering, Istocno Sarajevo, Bosnia and Hercegovina (e-mail: marijana.cosovic@etf.ues.rs.ba).

E. Junuz is with Dzemal Bijedic University, Faculty of Information Technology, Mostar, Bosnia and Hercegovina (e-mail: emina@edu.fit.ba).

datasets belong to different time periods. Data discretization was also performed to achieve better measures for evaluation of classification models. In addition to the original, normalized and discretized datasets data sampling algorithms were used. The use of sampling techniques in this paper aims at balancing the classes in anomalous datasets. Oversampling and undersampling algorithms were used as reported in [11], and measurements of the BGP anomaly classification ensemble models were compared, with or without data sampling techniques. BGP anomaly detection methodology is shown in Fig. 1. Further details can be found in [8].

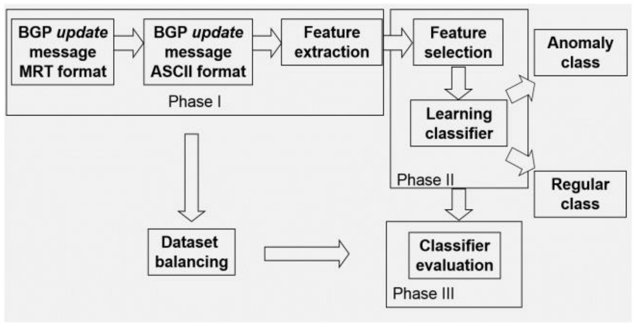

Fig. 1. BGP anomaly detection methodology [8].

III. Ensemble Classification Methods

By combining multiple classifiers ensemble methods are used to improve classification accuracy. Combined classifiers can improve accuracy [12] of individual models under several conditions: (i) individual model should classify a new instance with a classification error that is better than random errors and (ii) classification error should be of variable nature.

The pseudo code for the general procedure of the ensemble method is shown below. D is the original data set for training; N is the number of basic classifiers; T is a set of data for testing; and F is a function of the decision process, which defines the ensemble classifier. N different classifiers are formed based on N subsets of the original dataset D. On the test set, a classifier is formed in accordance with the decision process function.

```
for $i = 1$ to $N$ do
    form subset for training $D_i \in D$
    form basic classifier $h_i$
end for
for $\forall x \in T$ do
    $h^*(x) = F(h_1(x), h_2(x), \ldots, h_N(x))$
end for
```

The reasons why ensemble methods have more success than individual classifiers are statistical, computational and representational in nature. Risk of choosing an ineffective classifier is mediated by using a combination of classifiers and in this way various errors associated with single classifiers can be averaged out. All classifiers search the hypothesis space that can have infinitely many hypotheses so search process optimization becomes essential in determining an adequate hypothesis. Different techniques are used in case of inadequate datasets with either fewer or too many instances. Representational reasons are dealing with existence of an adequate and representative training dataset based on which the classifier can learn a dataset distribution.

A. Bagging

One of the basic techniques for ensemble methods construction is bagging [13]. It is based on creation of a large number of different datasets based on the original training dataset and combining them into one common model i.e. prediction based on the majority vote rule. New datasets are generated by random sampling of data with replacement from the original dataset i.e. using bootstrapping method. Every training dataset is independent and certain instances from the original dataset could occur several times in the new dataset, and some could be omitted. (Boot-strap AGGregatING).

The author in [14] has shown that slight changes in the training datasets when combined with weak classifiers create different hypotheses. Weak classifiers, such as decision trees and neural networks, are classifiers that for small changes in the training dataset exhibit changes in their classification performance. By using bagging technique, the variance shown in expression (1) can be reduced if the number of models m is high:

$$\sum_i \frac{(h^*(\mathbf{x}) - h_i(\mathbf{x}))^2}{m-1} \to 0 \tag{1}$$

B. Boosting

First effective boosting algorithm presented in several papers [15-17] used for binary classification is AdaBoost (Adaptive Boosting). The model, in general, is not taking independent random samples from the original training dataset but rather is generated in such a way that weights are successively applied to the training dataset.

The weights are increased or decreased in accordance with the experience of previous models, and a vote with a weight majority is applied in which more accurate models have greater influence in the voting [17]. Hence, a classifier which has an accuracy below 0.5 has a negative weight factor while a classifier which have accuracy greater than 0.5 receive positive weighting factors. In case of classifier which has an accuracy of 0.5, algorithm categorizes it as neutral in the final classifier outcome. AdaBoost algorithm adds models successively either until it generates a hypothesis that will perfectly classify the training dataset or until the predetermined maximum number of models is reached.

In the case of most learning algorithms, the error between the training and testing models increases with the complexity of the model. In the case of boosting, an increase of the number of iterations results in equalizing or even decreasing the difference between the model training and testing errors. This is due to the fact that the margin on a training dataset is higher and smaller errors are expected on the training dataset. Although the final classifier is seemingly complex, the increase in the margin reduction in the variance of the model is present [16].

The equation which represents the final classifier, and which is made of the K weak classifiers is given by:

$$H(x) = sign\left(\sum_{k=1}^{K} \omega_k h_k(x)\right) \tag{2}$$

where h_k is a hypothesis of the weak classifier, and ω_k

weighting factor as determined by the AdaBoost algorithm. Classifier error is calculated based on following equation

$$\omega_k = \frac{1}{2}\ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right) \tag{3}$$

where ε_k is the number of misclassified instances divided by the number of total instances in the training dataset. Equation (3) implies that in case of ε_k $\varepsilon_k \to 0$ we have that the weighting factor of the classifier increases exponentially, for a random guess $\varepsilon_k \to 0.5$ the weighting factor is zero, and for $\varepsilon_k \to 1$ the weighting factor decreases exponentially towards very large negative number. For calculated ω_k and in accordance with equation (4) we determine the weights for the i-th instance from the training dataset:

$$D_{k+1}(i) = \frac{D_k(i)exp(-\omega_k y_i h_k(x_i))}{Z_k} \tag{4}$$

where $D_k(i)$ is a vector of weighting factors used during the training of the k-th classifier. Each of the weight factors $D(i)$ represents the probability that the i-th instance would be selected as a part of the training dataset. Z_k is the sum of the weight factors and in the equation (4) is used as a normalizing factor such that:

$$Z_k = \sum_{1}^{n} D_k(i)exp(-\omega_k y_i h_k(x_i)). \tag{5}$$

h_k and y_i could have two possible values since it is a binary classification, hence their product could be -1 or 1. If the actual and predicted value is the same, their product is positive, and if they differ, the product is negative.

Outliers values that are not realistic could pose a problem to algorithm performance and hence it is advisable to remove these values from the training dataset prior to training the model. It is recommended for the training dataset using the AdaBoost algorithm to be of the best quality considering that ensemble methods in machine learning work on the correction of instances in the training dataset. Boosting methods are very sensitive to noise in the data, especially if it is in the output variables. Authors in [18] show that the AdaBoost algorithm overfits the data in the presence of noise in the training dataset therefore demonstrating the need for outlier removal. However, it has been shown in [19], that if model is based on a boosting method with an increased number of iterations it overfits much slower than a model based on other methods. The concept of soft margins also enrich the boosting method and it is used in RegBoost [20], Ada-BoostReg, Linear Programming Boost, and Quadratic Programming Boost [21] Same concept of soft margins and use of slack variable implementation for its regularization strategies is also applied in Support Vector Machines.

C. Random Forest

The first Random Forest algorithm, based on decision trees, has been developed in [22], [23]. Today's versions of the algorithm uses the bagging method of [13] together with [22], [24]. Authors in [25] show that AdaBoost could be considered a representative of Random Forest algorithms.

The advantage of usage of the decision tree is the speed of generating a classification model. On the other hand, the complexity of the model is limited and has the effect of degrading the accuracy of the training set. The Random Forest Method was created in response to improving accuracy with the increasing complexity of the model being trained. The basic principle on which this ensemble method is based, shown in Fig. 2, is that in a randomly selected space, which is a subset of the original space (D_1, D_2, .., D_N) of features, generates multiple decision trees T_1, T_2, .., T_N.

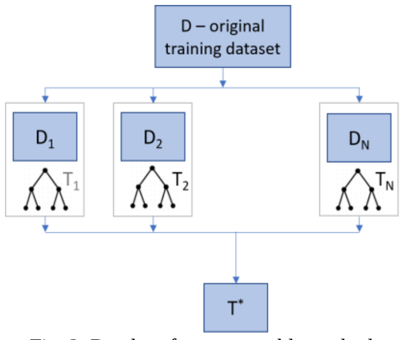

Fig. 2. Random forest ensemble method.

Bootstrap method has been used for generation of decision trees, which means that the trees are built from a subset of the original feature space with substitutions as shown in Fig. 2. The Random Forest method, unlike the decision trees that select the best feature for division of the feature space, uses k randomly chosen features for dividing the feature space. Typically for Random Forest $k = \sqrt{n}$, where n is the total number of features. Thus, a bagging method is combined with a randomly selected subset of features.

By combining decision trees generated in this way, classification accuracy increases, and it is associated with independence between the trees that make up the random forest. Also, the overfitting problem is minimized in the case of Random Forest method in comparison to the decision tree method. Random Forest is a method that can be implemented in parallel, and is also effective in dealing with a large number of input features, that is, with high-dimensional data, and the data in which distributions are not known.

IV. Results and Discussion

The AdaBoost algorithm, using the C4.5 decision tree for the base classifier, was developed in Python, and its performance measures are shown in Table I.

TABLE I: AdaBoost Method Performance Measures with C4.5 Decision Tree Base

Dataset	Acc	F-measure	MCC	ROC	PR	Time (s)
Slammer	0.9710	0.876	0.860	0.985	0.947	1.68
Nimda	0.7471	0.742	0.494	0.827	0.816	3.6
Code Red I	0.9457	0.639	0.614	0.870	0.645	3.04
Moscow PB	0.9967	0.927	0.926	0.977	0.932	1.31
AS9121 RTL	0.9986	0.938	0.937	0.999	0.976	0.65
Panix Hijack	0.9990	0.945	0.945	0.991	0.974	0.56
AS -PATH	0.9972	0.943	0.941	0.999	0.964	0.68
AWS	0.9949	0.823	0.825	0.957	0.835	1.44
Malay	0.9933	0.864	0.861	0.980	0.909	1.43
Indosat	0.9844	0.541	0.549	0.898	0.530	2.06

The classification is based on the combination of weak classifiers in order to create a strong one. Decision stump, the simplest form of decision tree, was used, and classification results improved in comparison to the C4.5 base algorithm. The basic function of the boosting algorithm is learning the weight factors of some weaker classifiers. The weight factors are adjusted based on the performance ratio of the new weak classifier and the previously trained ones. The training set adjusts in such a way that instances, which previous classifications have not successfully classified, have added weight factors, and for all other instances weight factors are reduced. We applied oversampling algorithms to Slammer, Code Red I, Moscow PB, and AS9121 RTL datasets. The best performance for AdaBoost classification is obtained when random oversampling (ROS) algorithm was applied on all discussed datasets. The random undersampling (RUS) algorithm applied to the Slammer and AS9121 RTL datasets gives the best classification performance when comparing all undersampling techniques. The Cluster Centroid (CC) algorithm applied to the Code Red I, and Moscow PB datasets has the best classification performance amongst ten undersampling techniques. Classification performance for Slammer, Code Red I, Moscow PB, and AS9121 has generally been improved by using undersampling and oversampling techniques compared to the original datasets, but oversampling techniques provide superior classification measures. Thus, for example, the number of incorrectly classified instances of anomaly is reduced from 254 to 11 for the Code Red I set. AdaBoost classification on oversampled Slammer, Moscow PB, and AS9121 RTL datasets has no incorrectly classified anomalous instances. The number of incorrectly classified regular instances increased using the oversampling technique for Slammer, Code Red I, and Moscow PB datasets while in the case of the AS9121 RTL dataset, the total number of incorrectly classified instances is lower than AdaBoost classification applied to the original datasets.

TABLE II: ADABOOST PERFORMANCE MEASURES WITH OVERSAMPLING/UNDERSAMPLING DATASETS

Dataset		F-measure	MCC	ROC	PR	Time (s)
Slammer	ROS	0.992	0.984	0.999	0.998	3.53
	RUS	0.949	0.898	0.989	0.990	0.39
Code Red I	ROS	0.985	0.969	0.997	0.995	5.61
	CC	0.884	0.765	0.948	0.938	0.34
Moscow PB	ROS	0.999	0.999	1	1	1.76
	CC	0.967	0.935	0.982	0.982	0.08
AS9121 RTL	ROS	0.999	0.999	1	1	1.32
	RUS	0.981	0.962	0.981	0.969	0.01

If we compare the execution time of the boosting ensemble method and the original datasets (Table I), as well as the oversampled datasets (Table II), we can conclude that the execution time is, as expected, longer than in the case of the oversampled datasets but approximately half as long for all original datasets apart in the case of the Moscow PB dataset. Since real-time anomaly detection requires a shorter model training time, the AdaBoost method with previously sampled data sets is adequate. Table III illustrates the performance of the bagging technique with a C4.5 decision tree which is used to implement a weak classifier. The bagging method generates multiple versions of the classifier that are used as a whole by the mechanism of the majority vote. Generation methods are based on manipulation of the training set, and it is recommended that the number of weak classifiers be limited to 10 [6] due to performance measure improvement.

TABLE III: BAGGING METHOD PERFORMANCE MEASURES WITH C4.5 DECISION TREE BASE

Dataset	Acc	F-measure	MCC	ROC	PR	Time (s)
Slammer	0.9732	0.855	0.871	0.986	0.950	0.75
Nimda	0.7581	0.754	0.516	0.842	0.835	2.4
Code Red I	0.9526	0.668	0.654	0.895	0.686	1.57
Moscow PB	0.9960	0.911	0.909	0.960	0.925	0.71
AS9121 RTL	0.9988	0.943	0.943	0.980	0.935	0.42
Panix Hijack	0.9993	0.962	0.962	0.998	0.950	0.65
AS -PATH	0.9974	0.945	0.944	0.995	0.954	0.31
AWS	0.9950	0.825	0.829	0.950	0.839	0.71
Malay	0.9928	0.853	0.850	0.980	0.907	0.63
Indosat	0.9861	0.565	0.588	0.913	0.609	0.81

The best classification performance for the bagging method is obtained for the ROS algorithm on all discussed data sets. The RUS algorithm applied to the Slammer and AS9121 RTL datasets gives the best classification performance when comparing all undersampling techniques. The CC algorithm applied to the Code Red I and Moscow PB datasets has the best classification performance of all ten undersampling techniques. As in the case of boosting, classification performance for all considered datasets has generally improved by using undersampling and oversampling techniques compared to original datasets, but oversampling techniques provide superior classification measures. The number of erroneously classified anomalous instances was reduced from 127 to 5 for the Slammer dataset and from 257 to 15 for the Code Red I dataset. On the other hand, by applying bagging techniques to the previously oversampled Moscow PB and AS9121 RTL datasets, there are no erroneously classified anomalous instances. The number of incorrectly classified regular instances increased using oversampling techniques for all datasets except for AS9121 RTL. The total number of incorrectly classified instances is lower than when the bagging method is applied to the original datasets.

TABLE IV: BAGGING PERFORMANCE MEASURES WITH OVERSAMPLING/UNDERSAMPLING DATASETS

Dataset		F-measure	MCC	ROC	PR	Time (s)
Slammer	ROS	0.989	0.977	0.999	0.999	2.17
	RUS	0.945	0.889	0.989	0.990	0.23
Code Red I	ROS	0.985	0.969	0.997	0.994	4.00
	CC	0.875	0.750	0.944	0.923	0.22
Moscow PB	ROS	0.999	0.998	0.999	0.999	1.49
	CC	0.961	0.923	0.981	0.972	0.04
AS9121 RTL	ROS	0.999	0.999	0.999	0.999	0.77
	RUS	0.981	0.962	1	0.999	0.01

If the execution time for the bagging ensemble method is compared to the original (Table III) and oversampled (Table IV) datasets, it can be concluded that the execution time is longer, as expected, in the case of oversampled datasets but less than two times as long in the case of Moscow PB and AS9121 RTL datasets and less than three times as long in the case of the remaining two datasets. Random forest classification algorithm, with a decision tree base classifier, was also implemented in Python. Table V shows the random forest measure performance with a C4.5 decision tree used as a weak classifier.

The best classification performance measures for random forest ensemble method were obtained by using the ROS algorithm on all considered datasets.

The RUS applied to the Slammer and AS9121 RTL datasets gives the best classification performance measures when comparing all undersampling techniques. The CC algorithm applied to the Code Red I and Moscow PB datasets has the best classification performance measures. Classification performance measures for all considered datasets generally improved by using oversampling and undersampling techniques compared to the original datasets, although oversampling techniques are superior in classification measures.

TABLE V: RANDOM FOREST METHOD PERFORMANCE MEASURES WITH C4.5 DECISION TREE BASE

Dataset	Acc	F-measure	MCC	ROC	PR	Time (s)
Slammer	0.9758	0.896	0.883	0.989	0.955	1.29
Nimda	0.7771	0.774	0.554	0.861	0.856	2.42
Code Red I	0.9528	0.664	0.653	0.901	0.694	1.75
Moscow PB	0.9960	0.910	0.909	0.979	0.933	1.32
AS9121 RTL	0.9988	0.944	0.944	0.994	0.983	0.69
Panix Hijack	0.9993	0.962	0.962	0.992	0.981	0.45
AS -PATH	0.9979	0.957	0.956	0.999	0.982	0.57
AWS	0.9951	0.826	0.832	0.970	0.864	1.17
Malay	0.9940	0.880	0.877	0.987	0.937	1.2
Indosat	0.9855	0.612	0.592	0.731	0.544	1.11

For example, the number of erroneously classified anomalous instances is reduced from 264 to 9 for the Code Red I set while using the random forest method on the oversampled Slammer, Moscow PB, and AS9121 RTL datasets causes no erroneously classified anomalous instances. The number of incorrectly classified regular instances increased using the oversampling technique in the Slammer, Code Red I, and Moscow PB datasets, while for the AS9121 RTL dataset, the total number of erroneously classified instances is lower in comparison to the random forest method applied to the original datasets.

If the execution time of the random forest ensemble method is compared to the original (Table V) and oversampled (Table VI) datasets it can be concluded that execution time is expectedly longer in the case of the oversampled datasets, and it is two times longer than in all datasets except the Moscow PB dataset for which the training time is one and a half times longer than the training time of the original dataset. Since real-time anomaly detection requires a shorter training time of the model this qualifies the random forest method with sampled datasets as an adequate choice.

Fig. 3. shows the execution time of three ensemble methods considered on original datasets. The bagging method has the shortest training time on all datasets except for the Panix hijack dataset in which the random forest method has shorter training time. The difference between the two models is minimal.

Fig. 4. shows the classifications performance measures (F-measure, MCC, ROC, and PR) for all three ensemble methods.

TABLE VI: RANDOM FOREST METHOD PERFORMANCE MEASURES WITH OVERSAMPLING/UNDERSAMPLING DATASETS

Dataset		F-measure	MCC	ROC	PR	Time (s)
Slammer	ROS	0.992	0.984	0.999	0.999	2.65
	RUS	0.954	0.908	0.991	0.990	0.28
Code Red I	ROS	0.989	0.979	0.999	0.999	3.48
	CC	0.883	0.767	0.950	0.944	0.32
Moscow PB	ROS	0.999	0.999	0.999	0.999	1.96
	CC	0.950	0.899	0.992	0.992	0.1
AS9121 RTL	ROS	0.999	0.999	0.999	0.999	1.51
	RUS	0.994	0.987	0.999	0.999	0.03

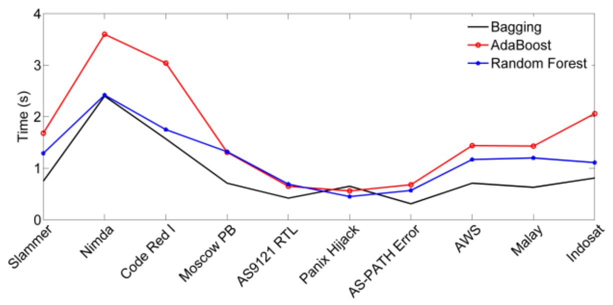

Fig. 3. Ensemble models execution time on considered datasets.

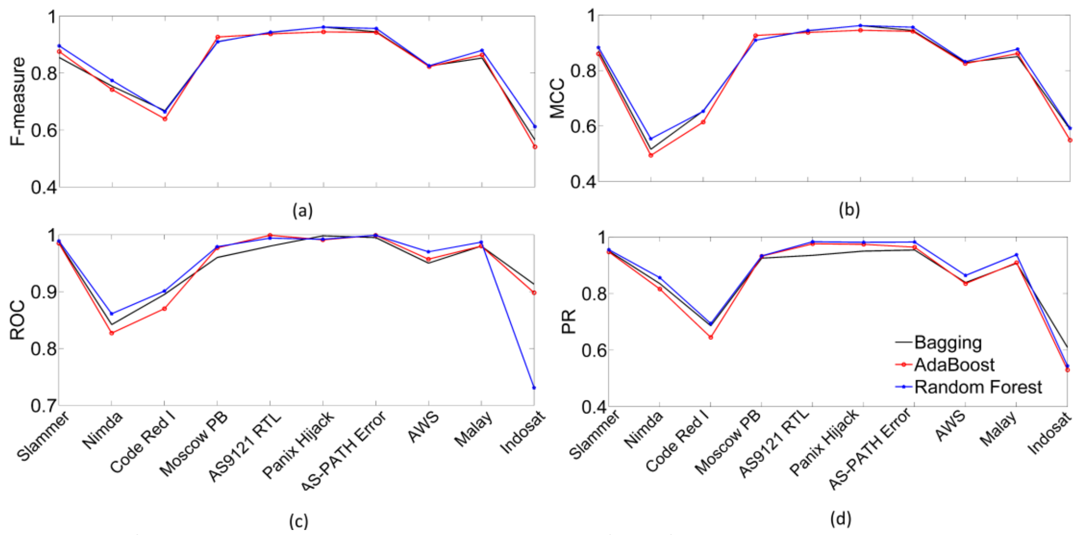

Fig. 4. Ensemble method performance measures (Bagging, AdaBoost, and Random forest) for Slammer, Nimda, Code Red I, Moscow PB, AS9121 RTL, Panix Hijack, AS-PATH Error, AWS, Malay, and Indosat datasets: (a) F-measure, (b) MCC, (c) ROC, and (d) PR.

V. Conclusion and Future Work

This paper compares several ensemble methods for improvement in performance measures in BGP anomaly detection domain. The random forest method compared to a single classifier models (SVM, NB and DT) provides the best tradeoff between model execution time and the rest of performance measures considered over all datasets. Bagging and boosting ensemble methods were used as well but the random forest ensemble method proved to be superior. For example, when random forest and bagging method were compared the biggest F-measure margin (4.7%) was obtained in case of Indosat amongst all datasets in the study. Slammer dataset has the next best F-measure value with a 4.1% margin while Malay dataset has a 3.7% margin when random forest method was applied in comparison to bagging. Remaining datasets have 2% or lesser margin of F-measure between the two compared methods. Boosting ensemble method was compared to the random forest as well.

We report better performance measures in case of the random forest when compared to boosting ensemble methods with greater margin than in case of bagging methods. For example, when random forest and boosting method were compared the biggest F-measure margin (7.1%) was obtained in case of Indosat amongst all datasets in the study. Nimda dataset has the next best F-measure value with a 3.2% margin while Code Red I dataset has a 2.5% margin when random forest method was applied in comparison to boosting. Remaining datasets have 2.3% or lesser margin of F-measure between the two compared methods.

We are planning to increase the number of anomalous datasets considered in this study and test the efficiency of the random forest method in the future.

References

[1] M. Cosovic, S. Obradovic, and L. Trajkovic, "Performance evaluation of BGP anomaly classifiers," in *Proc. Int. Conf. on Digital Inform., Networking and Wireless Commun.*, Moscow, Russia, Feb. 2015, pp. 115–120.

[2] M. Cosovic, S. Obradovic, and E. Junuz, "Deep Learning for Detection of BGP Anomalies," in *Proc. Int. Work-Conf. On Time Series*, Granada, Spain, Sept. 2017, pp. 487-498.

[3] M. Cosovic, S. Obradovic, and Lj. Trajkovic, "Classifying anomalous events in BGP datasets," in *Proc. 29th Annu. IEEE Can. Conf. on Electr. and Comput. Eng.*, Vancouver, Canada, May 2016, pp. 697–700.

[4] S. Nawata, M. Uchida, Y. Gu, M. Tsuru, and Y. Oie, "Unsupervised ensemble anomaly detection through time-periodical packet sampling," in *Proc. 13th IEEE Global Internet Symposium*, San Diego, CA, USA, March 2010, pp. 1-6.

[5] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, no. 3, pp. 249-268, 2007.

[6] J. Rodriguez, and J. Maudes, "Boosting recombined weak classifiers," *Pattern Recognition Lett.*, vol. 29, no. 8, pp. 1049-1059, 2008.

[7] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. on Document Anal. and Recognition*, Montreal, Canada, Aug. 1995, pp. 278–282.

[8] M. Cosovic, S. Obradovic, "Ensemble methods for classifying BGP anomalies," *Ind. Technolog.*, vol. 4, no. 1, pp. 12-20, June 2017.

[9] RIPE RIS raw data. [Online]. Available: http://www.ripe.net/data-tools/stats/ris/ris-raw-data

[10] University of Oregon Route Views project. [Online]. Available: http://www.routeviews.org/

[11] M. Cosovic and S. Obradovic, "BGP Anomaly detection with balanced datasets," *Tehnički vjesnik*, vol. 25, no. 3, pp. 766-775, June 2018.

[12] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. the 1st Int. Workshop on Multiple Classifier Systems*, London, UK, Aug. 2000, pp. 1-15.

[13] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, August 1996.

[14] R. E. Schapire, "The strength of weak learnability," *Mach. Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[15] R. E. Schapire, "Using output codes to boost multiclass learning problems," in *Proc. 14th Int. Conf. on Mach. Learning*, Nashville, TN, USA, July 1997, pp. 313-321.

[16] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," in *Proc. 14th Int. Conf. on Mach. Learning*, Nashville, TN, USA, July 1997, pp. 322-330.

[17] R. E. Schapire, and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proc. 11th Ann. Conf. on Comput. learning theory*, Madison, WI, USA, July 1998, pp. 80-91.

[18] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Mach. Learning*, vol. 42, no. 3, pp. 287-320, March 2001.

[19] P. Bühlmann, and B. Yu "Boosting with the L2 Loss," *Journal of the American Statistical Association*, vol. 98, no. 462, pp. 324-339, 2003.

[20] B. Kégl, and L. Wang, "Boosting on manifolds: adaptive regularization of base classifiers," in *Proc. 17th Int. Conf. on NEURAL Inf. Process. Syst.*, Vancouver, Canada, December 2004, pp. 665- 672.

[21] G. Rätsch, T. Onoda, and K. R. Müller, "Regularizing AdaBoost," in *Proc. Conf. on Neural Inf. Process. Syst.*, Denver, CO, USA, December 1998, pp. 564-570.

[22] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, Montreal, Canada, August 1995, pp. 278-282.

[23] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 20, no. 8, pp. 832-844, August 1998.

[24] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications*, R. Springer Publishing Company, Inc., 2014.

[25] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, no. 1, pp. 5-32, 2001.

Marijana Cosovic received her bachelor's degree in applied sciences from Simon Fraser University, Vancouver, Canada in 2005, M.Sc and Ph.D in computer science and telecommunications from University of East Sarajevo, B&H in 2010 and 2017, respectively. She is currently working as an assistant professor at the Faculty of Electrical Engineering, University of East Sarajevo, Bosnia and Hercegovina. Her research interest includes computational intelligence, machine learning and data analytics & optimization.

Emina Junuz received her bachelor's and master's degree in computer science engineering from University of Granada, Granada, Spain in 2003, and Ph.D in computer science from University Dzemal Bijedic, Mostar, Bosnia and Hercegovina in 2011. She is currently working as a professor at the Faculty of Information Technologies, Dzemal Bijedic University, Bosnia and Hercegovina. Her research interest includes databases, software engineering, information systems.