Journal of
Thai Interdisciplinary Research
Volume 13, Number 2, Pages 29 - 36

Applying an extremely imbalanced technique on big data: A case study of web intrusion

Kesinee Boonchuay[1,*], Sureerat Kaewkeeree[2], and Youppadee Intasorn[1]

[1]Computer Department, Faculty of Science and Technology, Songkhla Rajabhat University, Songkhla 90000, Thailand
[2]Business Computer Program, Faculty of Management Sciences, Songkhla Rajabhat University, Songkhla 90000, Thailand

Abstract

A web intrusion is a type of network intrusion that occurs frequently. A web log can be used to identify this type of intrusion. However, it tends to involve a huge amount of data which is difficult to be processed on a stand-alone computer. Moreover, this data is also an imbalanced dataset since the number of intrusion threats and normal accesses are extremely different. In order to handle a web intrusion, the two main topics that have to be involved include big data and an imbalanced problem. Therefore, this research applies an imbalanced technique based on big data to improve the performance on the web intrusion dataset. It is based on Apache Spark which is a popular open-source big data framework. The goal of this paper is to enhance the efficiency of intrusion prediction which is also categorized as an extremely imbalanced problem. The idea of minority class instance broadcasting is applied to improve the performance of prediction for web intrusion threats. According to the results, overall performance when applying an imbalanced technique with decision tree improves over a standard decision tree. For comparing by *F*-measure and geometric mean on 7 partitions, performances when applying an extremely imbalanced technique highly improve at 0.92 and 0.81 for *F*-measure and geometric mean respectively. For logistic regression, the application of an imbalanced technique does not show statistical improvement.

Keywords: classification, imbalance, big data, Apache Spark, web intrusion

Article history: Received 22 March 2018, Accepted 10 April 2018

1. Introduction

Nowadays, websites and internet applications have become part of everyday life. People exchange their public and personal data via internet without taking reasonable precautions. Due to the increase in client-server applications, the number of intrusion incidents is increasing gradually. Thailand Computer Emergency Response Team (ThaiCERT) [1] reported that intrusion attempts and intrusions occurred more than 1,105 times in 2017. Intrusion attempts accounted for 26.5% of incidents which was the largest number of security threat attacks in a single year.

In Thailand, it is prescribed by law to require the recording of all internet activities. In Songkhla Rajabhat University, our case study location, we collected all network traffic logs generated by the web server and examined them to find any suspicious behaviour. A web traffic log is considered an instance of big data. Its volume is large and real-time increasingly and needs the huge storage. Analysing a log file of such large size will consume all processors and resources. In this case, using a personal computer might not be suitable. Moreover, data collecting from a web traffic log is also considered an extremely imbalanced dataset because there are usually a minimal number of intrusion threats compared to normal accesses. Thus, in order to detect an attack incident, it is necessary to use a classification algorithm designed to deal with an imbalanced dataset. This leads to this research, which focuses on the experiment of web intrusion detection system. One general detection approach is to examine the log files or resources to detect threats and point out the bad incidents.

In this paper, we apply techniques based on big data to tackle the web intrusion dataset. For the big data, Apache Spark [2] is included in the research, which provides several libraries for big data especially machine learning library (MLlib) [3]. MLlib consists of many methods for classification such as decision tree, logistic regression, etc. To deal with an extremely imbalanced dataset, we apply the idea of broadcast minority class instances from EUS-ExtImbBD [4]. However, evolutionary under sampling (EUS) is omitted from this paper. Our assumption is only that the broadcast of minority class instances can improve the performance on accuracy of predicting an intrusion. In addition, this research aims to provide a simple solution to cope with extremely imbalanced big data as well as improve performance

*Corresponding author; e-mail: kesinee.bo@skru.ac.th

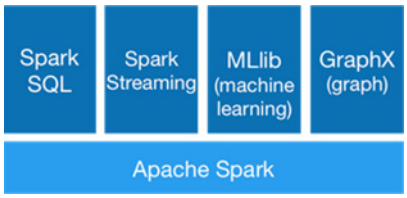


Figure 1 Apache Spark [2]

for the prediction of intrusion threats.

The paper is organized as follows. Section I is the introduction, while Section II is background. In Section III, the research methodology will be elaborated. In Section IV, the model evaluation and experimental results will be presented. The last section offers the conclusion and recommendations for future work.

2. Background and related work

In this section, the background topics related to this research are described. These topics include an imbalanced problem, classification in big data and techniques for handling imbalance in big data.

2.1. Classification in big data

Over the past few years, there has been increasing interest in big data. Such vast amounts of data were created every second, so the need for processing and analysing such data in quick and efficient ways becomes a serious problem. A number of tools and frameworks have been proposed to deal with big data problems, especially for Apache Hadoop and Apache Spark.

Apache Hadoop [5] is an open-source framework for distributing data to store on clusters. Hadoop has its own scalable file storage format called Hadoop Distributed File System (HDFS), which can store a very large data set separated on different clusters. It also has the MapReduce engine, a programming model that provides an environment to process large datasets with parallel and distributed algorithms. The Map phase will split the job into its computing nodes in order to process in parallel separately. In the Reduce phase, the master node will recall and merge all the small answers. According to its distributed infrastructure, all jobs will be processed and stored in its separated HDFS. Thus, the data transfer and performance of network interface can have an effect on the performance of the overall system.

Apache Spark is an open-source cluster processing framework for fast and flexible big data analysis. It has Resilient Distributed Dataset (RDD), which can perform in-memory processing. Apache Spark can work by relying on another cluster framework like Apache Hadoop or by just installing it as a stand-alone. It also comes up with efficient data analysis tools such as MLlib, Spark Streaming and GraphX.

In this research, we intend to use the fast in-memory processing ability of Apache Spark to help to process the log files.

2.2. An imbalanced problem

An imbalanced problem occurs when the input data of the classification method is not represented equally. The class of interest are typically smaller and might cause bias in the classification results. In this paper, the class having the larger number of instances is called the negative class, while the other class is called the positive class. This problem can be found frequently in the real world. However, most of traditional classifiers cannot handle this problem efficiently because they tend to predict instances as a negative class and misclassify positive class instances. Many researches have proposed techniques to solve this problem, which can be categorized into two main groups: data pre-processing techniques and algorithmic level techniques.

For pre-processing techniques, also known as sampling techniques, the idea of the techniques in this group involves increasing the number of positive class instances or decreasing the number of negative class instances, or both. These techniques attempt to balance the number of instances among classes. Examples of techniques in this group are SMOTE[6], Borderline-SMOTE [7], Safe-level SMOTE [8], ADASYN [9], MWMOTE [10], etc.

For algorithmic level techniques, they apply directly into the classifiers to compensate for the positive class instances. The classifiers can then enhance the performance of predicting positive class instances. Examples of techniques in this group are decision tree using asymmetric entropy [11], decision tree using off-centered entropy [12] and cost-sensitive boosting [13], among others.

2.3. Related works of handling imbalanced problem in big data

Imbalanced big data problems have received significant attention along with the growth of the data era. Therefore, many researches have proposed efficient modified algorithms to handle the imbalanced characteristics on big data. There are two popular strategies to cope with this problem: the application of random forest techniques and the use of sampling techniques.

For the first strategy, the idea of using multiple decision trees can gain advantages over distributed computing since each node can construct a small number of decision trees. The class having majority prediction from all decision trees is used as the prediction class for an instance. Therefore, these processes can be distributed to several computers to construct models and determine the prediction for each instance. The application of random forest for big data has been widely used in several researches such as [14] and [16]. In [14], Sarah Del Río et al. used a random forest classifier on MapReduce with a machine learning library on Hadoop called Mahout [15]. In [16], random forest was also implemented,

showing the success of using random oversampling on a big data problem.

For the second strategy, a sampling technique, especially under sampling, is one of the popular techniques used on big data. The idea of under sampling reduces the number of instances to be processed. Accordingly, it can improve the performance of processing time when compared with traditional techniques, which use all instances. An example of using under sampling to handle imbalanced problem can be seen in [4]. The author proposed EUS-ExtImbBD, which is based on the evaluation under sampling technique (EUS) [17]. EUS-ExtImbBD applies an evolutionary under sampling technique on big data using Apache Spark. This research focuses on an extremely imbalanced dataset, in which the imbalance ratio (IR) is very high. It uses the idea of broadcasting minority class instances to all partitions of all nodes instead of separating them to all partitions. Since the number of minority class instances is not reduced over multiple nodes, they can increase the performance on prediction of minority class instances in the distributed system. For using an under sampling technique, EUS-ExtImbBD can yield satisfactory processing time on big data.

In this research, we apply the idea of broadcast minority class instances from EUS-ExtImbBD to handle a web intrusion dataset. This dataset has a high imbalance ratio which is also an extremely imbalanced dataset. In order to provide a simple design for handling this type of dataset, MLLib of Apache Spark is used to construct models for classification instead of using EUS. By our assumption, using only broadcast minority class instances can improve the performance of prediction for intrusion threats in our web intrusion dataset.

3. Research methodology

For the web intrusion dataset, it can be tackled by using a traditional technique without applying big data. However, there are some limitations on computational performance which requires a high performance computer server to process. Moreover, it is hardly able to be integrated with an intrusion detection system in the future. Therefore, using big data technology, which provides a distributed computing environment, can eliminate this obstacle occurring in the standard technique. Apache Spark is used in this paper since it is one of the famous frameworks based on big data technology called Hadoop. The application of Apache Spark for the web intrusion dataset is described in the following steps.

3.1 Web intrusion dataset

In this step, the web log file from the date that has been informed as intrusions by the Office of Information Technology Administration for Educational Development [18] (called UniNet) about the intrusion is used for analysis. It is transformed into a dataset structure. The web log file consists of 8 parts comprising 1) IP address, 2) user name, 3) time-stamp, 4) access request, 5) status code, 6) the number of transferred bytes, 7) referrer URL and 8) user agent. An example of a web log file is below.

```
104.41.136.12 - - [06/Feb/2015:21:41:33 +0700] "GET /phpmyadmin/scripts/setup.php
HTTP=1:1" 404 304 "-" "Mozilla=5:0"
```

For the first transformation, the attributes in the dataset remain the same as in the web log file. Each record in the dataset is then labelled by an expert to indicate an intrusion threat. This dataset consists of 145,353 records, of which 144,841 records are normal accesses and 512 records are intrusions. The imbalance ratio (IR) of this dataset is 283.89, meaning it is an extremely imbalanced dataset. Therefore, it is suitable to handle using an imbalanced technique rather than a standard technique.

3.2 Data preparation

For data preparation, there are several transformations to prepare the dataset. First, an access request is transformed by whether it contains specific directory names or filename. Second, the location of an IP address is used to replace the IP address itself. Third, all categorical attributes are transformed by using one-hot encoding scheme in order to process by using MLlib in Spark. After these transformations, the size of this dataset will be much larger than the original. Its size is larger than one gigabyte. By applying it with big data, it can process this dataset easier than on a computer.

3.3 Feature selection

From the data preparation step, the number of attributes markedly increases. By using the large number of attributes, it consumes a lot of time to process the dataset. Moreover, it may lead to the data over fitting. Hence, chi-squared feature selection in MLlib is used to select important attributes that are relevant for the prediction of web intrusions. Therefore, there are 300 attributes in the dataset after this process.

3.4 Applying the imbalanced technique on Apache Spark

Since the web intrusion dataset used in the research is an extremely imbalanced dataset, the idea of broadcast minority class instances from EUS-ExtImbBD is applied. This design takes advantage of distributed computing to apply multiple nodes on multiple computers to process the task. In order to evaluate the models constructed by this process, k-fold cross-validation is included. The following algorithm shows the steps for processing this dataset on Apache Spark. There are two main functions of Apache Spark used in the algorithm which are map partition and broadcast. For map partition, it is applied in tree construction and model evaluation, which makes it able to distribute across multiple nodes. For broadcast, it prevents a decrease in the number of positive class instances over multiple partitions. The

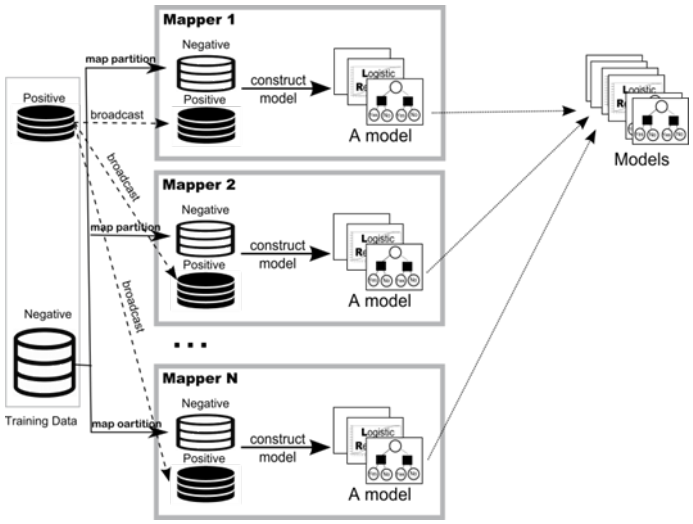


Figure 2 Training workflow

lower number of positive instances causes a classifier to be unable to predict a positive class instance correctly.

Algorithm 1 Extremely Imbalance Handling for a Web Intrusion Dataset

Input: 1) A dataset consists of positive instances (intrusion threats) and negative instances (normal accesses) and 2) number of partitions (#Partition)
Output: Results of model evaluation

```
1  set number of partition(#Partition) into data
2  for each  (trainFold, testFold) in data   //k-fold cross validation
3       posTrain = trainFold.filter("class== positive")
4       negTrain = trainFold.filter("class== negative")
5       posTrainBroadcast = broadcast(posTrain)
       //Training Process
6       map partition (negTrainPartition) for negTrain
7               model = ConstructModel(negTrainPartition,  posTrainBroadcast)
8               add  model to modelsPartition
9       modelsBroadcast = broadcast(modelsPartition)
       //Testing Process
10      map partition (testPartition) for testFold
11              evaluation =   EvaluateModel(testPartition,  modelsBroadcast)
12              add an evaluation to evaluationResults
13 return evaluationResults
```

This process can be split into two sub processes: training process and testing process. The workflow of the training process is shown in Figure 2. The training dataset is separated by class into two sets of instances: negative training instances and positive training instances. The negative training instances are split and distributed into each partition (mapper). Therefore, each partition contains a different set of negative training instances. For positive training instances, they are broadcasted to all partitions. Hence, all partitions have the same set of positive training instances. Each mapper constructs a model from the combined datasets of positive training instances and negative training instances in its partition. Then all models are aggregated together and used in the testing process.

In this research, MLlib is used to construct a model. There are three types of models: a decision tree, logistic regression and random forest. The details of model construction are shown in the following algorithm.

Algorithm 2 Construct Model

Input: 1) training positive instances (intrusion threats) and 2) training negative instances (normal accesses)
Output: a model

```
1  train = union( posTrain, negTrain )
2  create a model  from train by using MLlib in Spark
3  return a model
```

For the testing process, the testing dataset is split and distributed into each partition. Hence, each partition processes a different set of both positive class instances and negative class instances. All models which are the results from training process are broadcasted to all partitions.

For each partition, an instance is predicted by using a majority vote of all models. The results from the prediction are used to create confusion matrix. The models are evaluated by using accuracy, precision, recall, *F*-measure and geometric mean. The following algorithm shows the details for model evaluation.

Algorithm 3 Evaluate Model

Input: 1) testing instances (both positive and negative instances) and 2) models
Output: a result of model evaluation

```
1  predict all instances in testing data by using majority vote of  the results from all models
2  confusionMatrix = computeConfustionMatrix(prediction)
3  evaluationResult = evaluateModel(confustionMatrix)   //Evaluate by accuracy, F-measure, etc.
4  return evaluationResult
```

The detail for model evaluation and experiment results are described in the next section

4. Model evaluation and experimental results

4.1 Model evaluation

For model evaluation, they are evaluated by using 3-fold validation. The results are compared by using accuracy, precision, recall, *F*-measure and geometric mean. In order to compute these measures, the relevant terms have to be defined first which are true positive (***TP***), true negative (***TN***), false positive (***FP***) and false negative (***FN***). ***TP*** denotes the number of

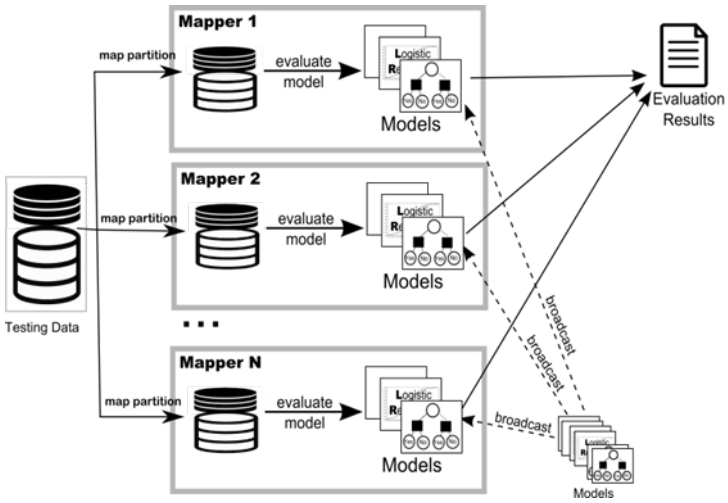


Figure 3 Testing workflow

positive class instances which are predicted as positive class instances. ***TN*** denotes the number of negative class instances which are predicted as negative class instances. ***FP*** denotes the number of negative class instances which are predicted as positive class instances. ***FN*** denotes the number of positive class instances which are predicted as negative class instances. In the Equation (1) and Equation (2), they present the formulae of precision and recall respectively.

$$\text{Precision} = \boldsymbol{TP} / (\boldsymbol{TP} + \boldsymbol{FP}) \quad (1)$$

$$\text{Recall} = \boldsymbol{TP} / (\boldsymbol{TP} + \boldsymbol{FN}) \quad (2)$$

In this paper, we concentrate on prediction of intrusion threats. Therefore, *F*-measure and geometric mean (g-mean) are suitable for our scenario. In Equation (3), it presents the formula of *F*-measure. In order to balance between precision and recall, the $\boldsymbol{\beta}$ is set as 1. For g-mean, it is defined by combining of precision and recall. Its formula is shown in the Equation (4).

$$F\text{-measure} = \frac{(1+\boldsymbol{\beta}^2)\cdot(\boldsymbol{Recall}\cdot\boldsymbol{Precision})}{\boldsymbol{\beta}^2\cdot\boldsymbol{Recall}+\boldsymbol{Precision}} \quad (3)$$

$$\text{Geometric mean (g-mean)} = \sqrt{\frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FN}}\times\frac{\boldsymbol{TN}}{\boldsymbol{TN}+\boldsymbol{FP}}} \quad (4)$$

The results comparing by using these measures are presented in the next section.

4.2 Experimental results

All processes are implemented by using Scala. The experiments run on a cloud service of Google cloud platform called Cloud Dataproc [19] which provides a service for Apache Spark and Hadoop. The service used in the experiments consists of 1 master node (8 vCPU and 52 GB memory) and 7 worker nodes (2 vCPU and 13 GB memory).

The experiment results in Figure 4 to Figure 6 present the results of using decision tree (abbreviated as DT), decision tree with imbalanced technique (abbreviated as IM_DT), linear regression (abbreviated as LR), linear regression with imbalanced technique (abbreviated as IM_LR) and random forest (abbreviated as RF). DT, IM_DT, LR and IM_LR are presented in a bar graph for comparing the results of the different number of partitions (1 to 7). For RF, the number of trees was fixed instead of the number of partitions. Since RF is created with a different setting, comparison with RF cannot be done directly. Therefore, the result of RF is presented in a different style (line graph).

For the first result in Figure 4, it presents a comparison of DT, IM_DT, LR, IM_LR and RF based on accuracy. It shows that the result of decision tree using imbalanced technique overcomes a standard decision tree especially in the number of partitions larger than one. For the result of logistic regression, it does not show significant difference overall between using standard logistic regression and logistic regression using imbalanced technique.

For the second result in Figure 5 (a), it shows the comparison of DT, IM_DT, LR, IM_LR and RF based on precision. It shows that the result of decision tree using the imbalanced technique overcomes a standard decision tree, especially when the number of partitions is larger than one. For the result of logistic regression, it does not show significant difference overall between using a standard logistic regression and logistic regression using the imbalanced technique. For the third result in Figure 5 (b), it shows the comparison by recall. Both decision tree and logistic

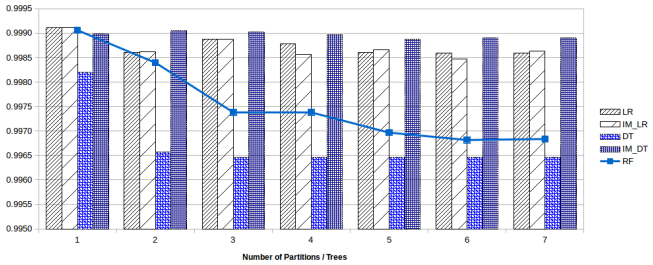
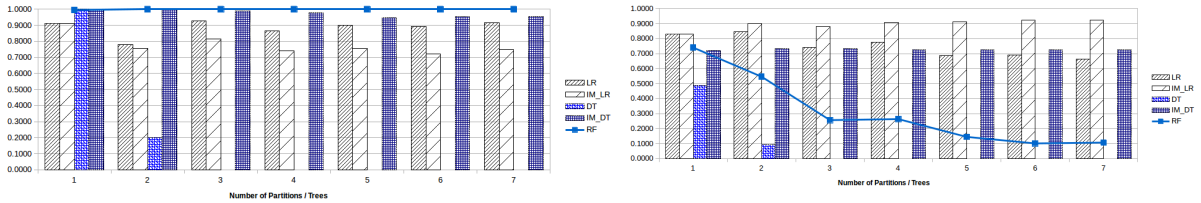
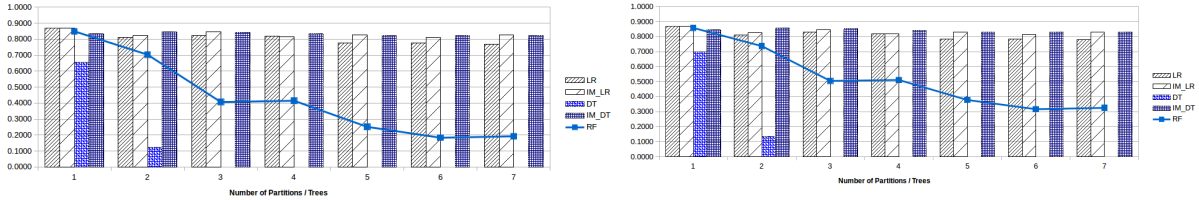


Figure 4 The experimental result comparing by accuracy



(a) precision (b) recall

Figure 5 The experimental result comparing by (a) precision and (b) recall



(a) *F*-measure (b) g-mean

Figure 6 The experimental result comparing by (a) *F*-measure and (b) g-mean

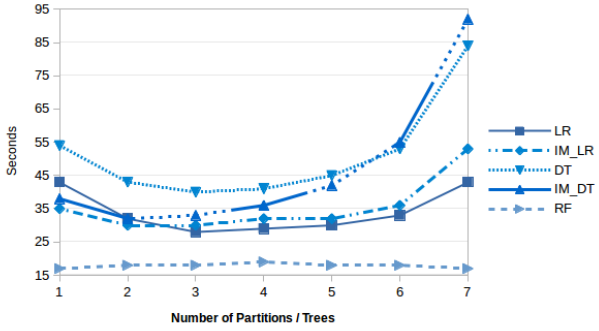


Figure 7 The experimental result comparing by processing times

regression using the imbalanced technique provide the improved performances compared to standard techniques.

The fourth and fifth results in Figure 6 show a comparison of DT, IM_DT, LR, IM_LR and RF by *F*-measure and g-mean. Both *F*-measure and g-mean provide similar results. The result of decision tree using the imbalanced technique is significantly enhanced compared to using a standard decision tree for any number of partitions. The results of logistic regression using the imbalanced technique is slightly improved compared to standard logistic regression.

The sixth result in Figure 7 presents a comparison of processing times for DT, IM_DT, LR, IM_LR and RF. The processing times for all seem to provide a minimum processing time at around 2-3 partitions. The result of decision tree using the imbalanced technique consumes less overall processing time than a standard decision tree. The result of logistic regression using the imbalanced technique shows slightly higher

processing time is used compared to a standard logistic regression.

Overall, the result of decision tree using the imbalanced technique is significantly improved over a standard decision tree as compared by *F*-measure and g-mean, which are widely used for imbalanced datasets. The processing time for decision tree using the imbalanced technique also improves when compared to a standard decision tree. For logistic regression, using the imbalanced technique does not seem to significantly enhance the result as compared by all measures. Therefore, positive instance broadcasting alone is probably not sufficient for improving an extremely imbalanced dataset using logistic regression. For random forest, it applies sampling methods which yield the least processing time compared with others. It also trades off performance with performance on accuracy, precision, recall, *F*-measure and g-mean.

6. Conclusion and future work

In this paper, big data is applied with a web instruction dataset to improve the performance of classification. Since the preparation process increases the size of the dataset to more than a gigabyte, it is difficult to process this task on a stand-alone computer. Therefore, Apache Spark based on big data is used to handle this dataset. In addition, this dataset is also extremely imbalanced. This research applies the use of MLlib on Apache Spark to improve performance for the prediction of an imbalanced dataset. The idea of broadcasting positive instances from EUS-ExtImbBD is used with MLlib to handle the web instruction dataset.

According the results, a decision tree using an imbalanced technique shows significant overall improvement. Therefore, this technique is suitable for a decision tree. However, it does not show strong improvement for logistic regression. To deal with an extremely imbalanced dataset using logistic regression, integrating other techniques with positive instance broadcasting is probably required. In conclusion, the technique presented in this paper can be applied effectively with decision tree to improve prediction for an extremely imbalanced dataset.

In future work, we aim to integrate this technique to a real-time system. Therefore, the application of a sampling method has to be included to reduce processing time.

Acknowledgements

The authors would like to acknowledge the Faculty of Science and Technology, Songkhla Rajabhat University, Thailand for supporting this research. We also thank the computer center of Songkhla Rajabhat University for providing log files.

References

[1] Thailand Computer Emergency Response Team (ThaiCERT) n.d. http://www.thaicert.or.th (accessed May 18, 2017).

[2] Spark A. Apache Spark: Lightning-fast cluster computing. Retrieved from Apache Spark: http://spark. apache. org; 2016.

[3] Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, et al. Mllib: Machine learning in apache spark. J Mach Learn Res 2016;17:1235–1241.

[4] Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F. Evolutionary undersampling for extremely imbalanced big data classification under apache spark. Evol. Comput. CEC 2016 IEEE Congr. On, IEEE; 2016.p.640–647.

[5] Apache Hadoop. Apache Hadoop 2017. http://hadoop.apache.org/.

[6] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. J Artif Int Res 2002;16:321–357.

[7] Han H, Wang W-Y, Mao B-H. Borderline-SMOTE: A new Over-Sampling method in imbalanced data sets learning. In: Huang D-S, Zhang X-P, Huang G-B, editors. Adv. Intell. Comput. Int. Conf. Intell. Comput. ICIC 2005 Hefei China August 23-26 2005 Proc. Part I, Berlin, Heidelberg: Springer Berlin Heidelberg; 2005.p. 878–87. doi:10.1007/11538059_91.

[8] Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C. Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling Technique for handling the class imbalanced problem. In: Theeramunkong T, Kijsirikul B, Cercone N, Ho T-B, editors. Adv. Knowl. Discov. Data Min., vol. 5476, Springer Berlin Heidelberg; 2009.p. 475–82.

[9] He H, Bai Y, Garcia EA, Li S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. Neural Netw. 2008 IJCNN 2008 IEEE World Congr. Comput. Intell. IEEE Int. Jt. Conf. On, 2008. p. 1322–8.

[10] Barua S, Islam MM, Yao X, Murase K. MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 2014;26:405–425.

[11] Marcellin S, Zighed DA, Ritschard G. An asymmetric entropy measure for decision trees 2006.p. 1292–9.

[12] Lenca P, Lallich S, Vaillant B. Construction of an Off-Centered entropy for the supervised learning of imbalanced classes: Some First Results. Commun Stat - Theory Methods 2010;39:493–507.

[13] Sun Y, Kamel MS, Wong AK, Wang Y. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit 2007;40:3358–3378.

[14] Del Río S, López V, Benítez JM, Herrera F. On the use of MapReduce for imbalanced big data using Random Forest. Inf Sci 2014;285: 112–137.

[15] Lyubimov D, Palumbo A. Apache Mahout: Beyond MapReduce. CreateSpace Independent Publishing Platform; 2016.
[16] Triguero I, del Río S, López V, Bacardit J, Benítez JM, Herrera F. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: an extremely imbalanced big data bioinformatics problem. Knowl-Based Syst 2015;87:69–79.
[17] García S, Herrera F. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. Evol Comput 2009;17:275–306.
[18] UniNet. Off Inf Technol Adm Educ Dev 2017. http://www.uni.net.th/.
[19] CLOUD DATAPROC. Cloud Datapro - Cloud-Native Hadoop Spark 2017. https://cloud.google.com/dataproc/.