Mathematics and Statistics 9(3): 379-385, 2021 http://www.hrpub.org
DOI: 10.13189/ms.2021.090320

Comparing the Performance of AdaBoost, XGBoost, and Logistic Regression for Imbalanced Data

Sharmeen Binti Syazwan Lai, Nur Huda Nabihan Binti Md Shahri*, Mazni Binti Mohamad, Hezlin Aryani Binti Abdul Rahman, Adzhar Bin Rambli

Centre of Statistical and Decision Sciences Studies, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Malaysia

Received November 17, 2020; Revised March 9, 2021; Accepted March 21, 2021

Cite This Paper in the following Citation Styles

(a): *[1] Sharmeen Binti Syazwan Lai, Nur Huda Nabihan Binti Md Shahri, Mazni Binti Mohamad, Hezlin Aryani Binti Abdul Rahman, Adzhar Bin Rambli , "Comparing the Performance of AdaBoost, XGBoost, and Logistic Regression for Imbalanced Data," Mathematics and Statistics, Vol. 9, No. 3, pp. 379 - 385, 2021. DOI: 10.13189/ms.2021.090320.*

(b): *Sharmeen Binti Syazwan Lai, Nur Huda Nabihan Binti Md Shahri, Mazni Binti Mohamad, Hezlin Aryani Binti Abdul Rahman, Adzhar Bin Rambli (2021). Comparing the Performance of AdaBoost, XGBoost, and Logistic Regression for Imbalanced Data. Mathematics and Statistics, 9(3), 379 - 385. DOI: 10.13189/ms.2021.090320.*

Copyright©2021 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract An imbalanced data problem occurs in the absence of a good class distribution between classes. Imbalanced data will cause the classifier to be biased to the majority class as the standard classification algorithms are based on the belief that the training set is balanced. Therefore, it is crucial to find a classifier that can deal with imbalanced data for any given classification task. The aim of this research is to find the best method among AdaBoost, XGBoost, and Logistic Regression to deal with imbalanced simulated datasets and real datasets. The performances of these three methods in both simulated and real imbalanced datasets are compared using five performance measures, namely sensitivity, specificity, precision, F1-score, and g-mean. The results of the simulated datasets show that logistic regression performs better than AdaBoost and XGBoost in highly imbalanced datasets, whereas in the real imbalanced datasets, AdaBoost and logistic regression demonstrated similarly good performance. All methods seem to perform well in datasets that are not severely imbalanced. Compared to AdaBoost and XGBoost, logistic regression is found to predict better for datasets with severe imbalanced ratios. However, all three methods perform poorly for data with a 5% minority, with a sample size of *n* = 100. In this study, it is found that different methods perform the best for data with different minority percentages.

Keywords Imbalanced Data, Adaboost, XGBoost, Logistic Regression and Performance Measure

1. Introduction

Statisticians often encounter the imbalanced data issue when performing classification tasks. Imbalanced data, which is the major source of classification problems, normally occurs when the distribution between outcome classes is not equal [1]. Imbalanced data happens when there is one majority class that makes up almost the whole data and one small minority class [2]. In imbalanced data, the minority class always contains important information. Hence, standard classification is not the best approach to predictions involving imbalanced datasets [3].

Real world applications of classifications that involve imbalanced data include text categorizing, fault and fraud detection, and oil spill detection through satellite images [1]. It is crucial to find the best way to deal with imbalanced datasets [4]. [5] emphasizes finding a solution for an accurate prediction of the minority cases.

Many issues arise in mining imbalanced cases. The first issue is classification accuracy. Even though the evaluation metrics method is commonly used for the classification task, it is not appropriate for classifying imbalanced cases because it is a biased performance measure [6]. The second issue in mining imbalanced cases is a lack of data (absolute rarity). This problem arises when the number of observations in the minority class is small in the absolute sense, making it almost impossible for the classifier to detect any regularities in the minority class [6]. The third issue is a relative lack of data (relative rarity). This problem arises when the minority cases are not rare in an absolute sense but are rarely relative to other cases, rendering it difficult to identify patterns [6]. The fourth issue is data fragmentation; partitioning the data into smaller groups can also lead to an absolute lack of data within a single partition [6]. Fifth, noise affects the minority class more than the majority class and has a large negative impact on detecting the minority cases, making it difficult to form decision boundaries around the minority class [6]. Lastly, inappropriate inductive bias can affect the overall ability to learn about the minority class [6].

Machine learning algorithm is always preferred than traditional classification approaches when it comes to handling imbalanced data. Traditional classification methods tend to ignore the imbalanced data issue, leading to poor classification performance in imbalanced datasets. Besides, the imbalance problem is associated with the ratio of the majority to the minority class, the overall training data set, the classifier involved, and the complexity concept indicated by the data [7].

Thus, this study intends to investigate the performance of two boosting algorithms, namely AdaBoost and XGBoost, along with a regression method known as logistic regression.

2. Methodology

2.1. Adaptive Boosting (AdaBoost)

AdaBoost [8] is the earliest boosting technique introduced by Freund and Schapire [9]. AdaBoost makes use of weak classifiers to build a stronger and more stable classifier. According to Tavish [10], the initial step is where the base learner or the weak learner allocates equal weight or attention to each observation. Once weight has been assigned to each observation, the weak learner can be used for prediction. Greater weights are then assigned to observations that are misclassified by the weak learner, and the next base learner is used for prediction. This process will be iterated until it reaches the limit of the base learning algorithm, T_i, where the iteration is denoted as t. In the final step, the outputs of the weak learner will be combined to develop a stronger learner that will enhance the prediction accuracy. In this study, the decision tree is chosen as the weak classifier with a depth of 1.

According to Freund and Schapire[11], the AdaBoost algorithm takes places in the training set, which contains the vectors (x_1, y_1), …, (x_k, y_k), where each of x_k is an element from the domain X, while each of y_k is an element from the set Y. In this study, the labels in Y are set as $\{0,1\}$, where 0 is for the majority class and 1 is for the minority class in each dataset. At each iteration t, the weight of the distribution attributed to case i, denoted as $D_t(i)$ on $\{1,2,\ldots,k\}$, is calculated. In the first iteration where $t=1$, the weight is set equally, $D_1(i)=1/k$. Next, a weak classifier, $h_t(x) \to X\{0,1)$ is obtained. The error made by the weak learner $h_t(x)$ is then calculated by $\varepsilon_t = \sum_{i=1}^{k} D_t(i) * \delta_i$, where δ_i is 0 if $h_t(x_i) = y_i$ and 1 otherwise. At iteration $t=2$, the weight is updated by $D_{t+1}(i) = D_t(i) * F_i / Z_t$.
where,

$$\boldsymbol{F_i} = \boldsymbol{e^{-\alpha_t}} \text{ if } \boldsymbol{h_t(x_i) = y_i};\ \boldsymbol{e^{\alpha_t}} \text{ if } \boldsymbol{h_t(x_i) \neq y_i} \quad (1)$$

$$\boldsymbol{\alpha_t = 0.5 \ln(\frac{1-\varepsilon_t}{\varepsilon_t})} \quad (2)$$

Z_t = normalisation constant such that $\sum_{i=1}^{k} D_t(i) = 1$

2.2. Extreme Gradient Boosting (XGBoost)

Boosting involves assembling all the weak classifiers to produce a strong classifier. XGBoost, a form of boosting, was developed from the existing Gradient Boosting technique. According to [12], XGBoost was adapted from Gradient Boosting to enhance the computing speed, scalability, and generalization performance. The first step in implementing XGBoost is to organize the data. All the categorical forms of the data will be transformed into the numeric form since XGBoost only works with numeric vectors. This transformation can be completed using One Hot Encoding. The next step is to clean the data and undergo feature engineering.

The estimated model can be obtained using the general function, as stated in the following equation:

$$\boldsymbol{\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)} \quad (3)$$

where,
$\hat{y}_i^{(t)}$ = predictions at step t
$f_t(x_i)$ = the learner at step t
x_i = the input variable
$\hat{y}_i^{(t-1)}$ = predictions at step $t-1$

The aim is to stop overfitting in XGBoost, as shown in the equation below:

$$\boldsymbol{F_{obj}(\theta) = \sum_{k=1}^{n} L(\theta) + \sum_{k=1}^{t} \Omega(\theta)}, \quad (4)$$

where,
$\boldsymbol{L(\theta) = l(\hat{y}_i, y_i)}$, a loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i
$\boldsymbol{\Omega(\theta) = \gamma T + \frac{1}{2}\lambda||w||^2}$, a regularized term that penalizes complex models
$\gamma =$ the min. loss needed to further partition the leaf note
$T =$ the number of leaves in the tree
$\lambda =$ a regularized parameter to scale the penalty
$\gamma T =$ spanning the tree pruning
$w =$ weight of leaves

Then, a second order expansion is taken to the loss function in XGBoost, and the final aim is shown in equation (5),

$$\boldsymbol{J^{(t)}} \approx \sum_{i=1}^{n}\left[\boldsymbol{g_i w_{q(x_i)}} + \frac{1}{2}\left(\boldsymbol{h_i w_{q(x_i)}^2}\right)\right] + \boldsymbol{\gamma T} + \frac{1}{2}\boldsymbol{\lambda}\sum_{j=1}^{T}\boldsymbol{w_j^2} \quad (5)$$

where,

h_i = the second derivative of the loss function

g_i = the first derivative

The optimal weight of leaf j, w_j can be identified using the following equation,

$$\boldsymbol{w_j} = -\frac{\sum_{i\epsilon I_j}\boldsymbol{g_i}}{\sum_{i\epsilon I_j}\boldsymbol{h_i}+\boldsymbol{\lambda}} \quad (6)$$

The sum of loss value for every leaf node can be characterized by the upcoming loss function in the equation

$$\boldsymbol{J^{(t)}} \approx \sum_{j=1}^{T}\left[\left(\sum_{i\epsilon I_j}\boldsymbol{g_i}\right)\boldsymbol{w_j} + \frac{1}{2}\left(\sum_{i\epsilon I_j}\boldsymbol{h_i} + \boldsymbol{\lambda}\right)\left(\boldsymbol{w_j^2}\right)\right] + \boldsymbol{\gamma T} \quad (7)$$

where I_J refers to all data samples in the leaf node j. Thus, the change in the model's performance can be assessed based on the objective function, after a node split in the decision tree [13].

2.3. Logistic Regression

Logistic regression [14] is a regression technique used when the response variable is a binary, mutually exclusive variable. In other words, logistic regression is used for modelling binary outcomes, which would involve an estimation task because it tries to answer the probability of a record belonging to a relevance class. The response variable has only two possible outcomes, which can be represented by a binary indicator variable taking the values of 0 and 1. A logistic regression model is made up of one or more independent variables (continuous or categorical) and one categorical dependent variable. Given p is the probability of success, p is transformed into odds to form a linear equation of an exponential function form,

$$\boldsymbol{p} = \mathbf{exp}(\boldsymbol{\beta_0} + \boldsymbol{\beta_1 x_1} + \cdots + \boldsymbol{\beta_n x_n}) \quad (8)$$

where p is expressed in terms of odds,

$$\boldsymbol{Odds} = \frac{\boldsymbol{p}}{\mathbf{1}-\boldsymbol{p}} = \mathbf{exp}(\boldsymbol{\beta_0} + \boldsymbol{\beta_1 x_1} + \cdots + \boldsymbol{\beta_n x_n}) \quad (9)$$

Taking the natural log on the both sides, the logit function will be as follows,

$$\mathbf{log}\left(\frac{\boldsymbol{p}}{\mathbf{1}-\boldsymbol{p}}\right) = \boldsymbol{\beta_0} + \boldsymbol{\beta_1 x_1} + \cdots + \boldsymbol{\beta_n x_n} \quad (10)$$

where,

β_0 = constant

β_n = coefficient of the independent variable x_n

2.4. Performance Measure

A confusion matrix will be used to evaluate the output of the approaches by documenting how many instances are correctly and incorrectly found. Next, the efficiency of the three classifiers will be determined based on their sensitivity, specificity, and g-mean. While sensitivity is the percentage of real positives that are correctly recognised, specificity is the percentage of real negatives that are correctly recognised.

2.5. Simulation Study

A Monte Carlo simulation study is performed to generate nine imbalanced datasets having different imbalance ratios of 5%, 10%, and 25% with different sample sizes of *n* = 100, 300, and 500. This study adapted a simulation study done by [15], where the imbalanced data is simulated to suit a binary logistic regression with x_1 and x_2 as the independent variables and y as the binary dependent variable. In this simulation study, the coefficient values of β_0 and β_1 are set at 1.08 and 2.08, respectively, similar to the values chosen by Rahman and Yap [15] in their study. The dependent variable, y is a binary variable with an assigned value of 0 or 1. Cases with a probability of less than or equal to 0.5 generated by the model will fall into class 0, while the ones having the probability of more than 0.5 will fall into class 1. The looping will stop upon reaching the imbalance ratio that has been set.

The simulation process is replicated for about 5,000 times with different imbalance ratios and sample sizes. Each generated data is partitioned into two sets, with 70% as the training set and 30% as the testing set. The training sets are then used to model AdaBoost, XGBoost, and logistic regression; hence, these models are used for prediction purposes. The performance of each model is assessed using both training and test sets. The mean values of sensitivity, specificity, precision, F1-score, and g-mean are obtained to assess the performance of each classifier.

2.6. Application of Real Datasets

The data used to achieve the second objective is secondary data instead of primary data. The secondary data was retrieved from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.html), which contains numerous collections of datasets created by students of UC Irvine.

In this study, the selected datasets are Glass, Ecoli, and Wifi Localization from UCI Repository. Three degrees of imbalance are tested, namely 5% (highly imbalanced), 10% (moderately imbalanced), and 25% (imbalanced). These imbalanced datasets are then compared to see how they affect the training of the classifiers.

3. Results

This section presents the analyses and findings of AdaBoost, XGBoost, and logistic regression on the simulation study, the descriptive analysis of the imbalanced real datasets, and the performance of each classifier model for each imbalanced real dataset. These analyses are conducted using R Statistical Software version 3.5.1. All three methods perform the worst with the dataset of 5% minority class and 500 sample size. We can conclude that as the sample size increases and the percentage of the minority class decreases, the classifiers fail to predict the minority cases as the classifiers treat the minority cases as noise. Thus, the classifiers focus on predicting the majority cases, with good outcomes. Logistic regression performs the best for most of the datasets, and it outperforms the boosting methods for the datasets with a 5% minority class.

3.1. Simulated Imbalanced Datasets

3.1.1. Minority

Table 1. Minority Percentage Results

Minority	
5%	A constant pattern is observed for all sample sizes, as the specificity rate seems to be very high across all three sample sizes. Besides, it seems that the sensitivity of AdaBoost and XGBoost does not change much across all three sample sizes. Hence, sample size does not influence the sensitivity of AdaBoost and XGBoost. A larger sample size is needed to obtain readable results for AdaBoost and XGBoost. Meanwhile, logistic regression shows a constant pattern for all sample sizes used.
10%	A constant pattern is observed across all sample sizes. Specificity seems to be very high across all three sample sizes. Meanwhile, sensitivity is very low for all sample sizes and is not as high as specificity. It seems that the precision of AdaBoost, XGBoost, and logistic regression is the lowest when the sample size is 100, and the precision of logistic regression is the lowest when the sample size is 300. However, when the sample size is increased to 500, both methods show an increment in precision. As for the F1-score and g-mean, logistic regression shows a systematic improvement as the sample size increases. Hence, logistic regression is the most stable for the datasets with a 10% minority class.
25%	A constant pattern is observed for all sample sizes. Specificity seems high across all three sample sizes. Besides, all the sample sizes show relatively high sensitivity. The F1-score and g-mean are the highest when the sample size is 100 for all three methods.

3.1.2. Methods

Table 2. Performance of Methods

Methods	
AdaBoost	AdaBoost performs well when the imbalance is not severe. For instance, the sensitivity rate is the highest, at about 64.58% for the dataset with a 25% minority class and a sample size of 500. AdaBoost's performance improves as the minority percentage increases. In addition, AdaBoost's F1-score and g-mean also increase as the sample size increases to 500. AdaBoost has the best sensitivity, F1-score, and g-mean for the dataset with a sample size of 300 and 25% minority. On the contrary, it has the lowest sensitivity, F1-score, and g-mean when the sample sizes are 100 (with 25% minority) and 300 (with 10% minority).
XGBoost	XGBoost performs the best when the imbalance is severe, which is when n = 100 with 25% minority, achieving the highest sensitivity rate of 71.85%. This is followed by 56.75% sensitivity when n = 500 with 25% minority. XGBoost's performance is inconsistent as the g-mean does not show a consistent improvement as the sample size increases.
Logistic regression	Logistic regression performs the best when the imbalance ratio is not severe. It achieves the highest sensitivity rate of 65.78% when n = 100 with 25% minority. Logistic regression also achieves a relatively high g-mean and F1-score with the 10% minority datasets. Logistic regression performs better that boosting algorithm for most datasets, such as the 10% and 25% minority datasets.

3.1.3. Performance Measure

Table 3. Results of Performance Measure

Performance Measure	
Sensitivity	All three methods have the lowest sensitivity rate when the dataset is highly imbalanced, with a 5% minority class in a sample size of 100. All three methods start to improve, showing good performance in predicting positive classes when the datasets have a sample size of 500 onwards with a 5% minority class.
Specificity	All three methods show very consistent high specificity rates for all minority percentages and all sample sizes. However, the sensitivity rate for all three methods is the lowest when the sample size is set at 100 with a 25% minority class.
Precision	All three methods have the lowest precision rate when the minority class is highly imbalanced at 5% and the sample size is set at 500. All three methods start to improve, showing good performance in predicting positive cases when the dataset is not highly imbalanced (10%) and the sample size is set is 500 onwards.
F1-score	All three methods have the lowest F1-score when the minority class is highly imbalanced at 5% and the sample size is set at 100, except for logistic regression. All three methods start to improve, showing good performance in predicting positive cases when the dataset is set at 5% minority with a sample size of 500 onwards.
G-mean	All three methods have the lowest g-mean when the minority class is highly imbalanced at 5% with a sample size of 100. All three methods start to improve, showing good performance in predicting positive cases when the dataset is set at 5% minority with a sample size of 500 onwards.

3.2. Real Imbalanced Datasets

3.2.1. Glass Dataset

Table 4. Performance Results of Glass Dataset

Glass	AdaBoost		XGBoost		Logistic	
Dataset	Train	Test	Train	Test	Train	Test
TN	144	60	144	61	144	60
FN	0	0	0	0	0	0
FP	0	1	0	0	0	1
TP	7	2	7	2	7	2
Sensitivity	1.000	1.000	1.000	1.000	1.000	1.000
Specificity	1.000	0.983	1.000	1.000	1.000	0.983
Precision	1.000	0.666	1.000	1.000	1.000	0.666
F1-Score	1.0000	0.8000	0.5714	1.0000	1.0000	0.8000
g-mean	1.0000	0.9918	1.0000	1.0000	1.0000	0.9918

3.2.2. Ecoli Dataset

Table 5. Performance Results of Ecoli Dataset

Ecoli	AdaBoost		XGBoost		Logistic	
Dataset	Train	Test	Train	Test	Train	Test
TN	211	87	211	84	205	86
FN	4	2	0	2	13	3
FP	0	3	0	6	6	4
TP	21	8	25	8	12	7
Sensitivity	0.8400	**0.8000**	1.0000	**0.8000**	0.4800	0.7000
Specificity	1.000	**0.966**	1.000	0.933	0.971	0.955
Precision	1.000	**0.727**	1.000	0.571	0.666	0.636
F1-Score	0.913	**0.761**	0.571	0.666	0.558	0.666
g-mean	0.916	**0.879**	1.000	0.864	0.682	0.817

3.2.3. Wifi Dataset

Table 6. Performance Results of Wifi Dataset

Wifi	AdaBoost		XGBoost		Logistic	
Dataset	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
TN	1049	447	1050	447	1003	433
FN	0	2	0	1	84	37
FP	1	3	0	3	47	17
TP	350	148	350	149	266	113
Sensitivity	0.8800	0.9867	1.0000	**0.9933**	0.7600	0.7533
Specificity	0.9990	**0.9933**	1.0000	**0.9933**	0.9552	0.9622
Precision	0.9972	0.9801	1.0000	**0.9803**	0.8498	0.8692
F1-Score	0.9986	0.9834	0.5714	**0.9868**	0.8024	0.8071
g-mean	0.9995	0.9900	1.0000	**0.9933**	0.8520	0.8514

Regarding the imbalanced real datasets, all the methods perform perfectly in predicting the positive class in the glass dataset, achieving a 100% rate for sensitivity. However, there is an overfitting issue in the case of AdaBoost and logistic regression, as the training set obtain better results than the testing set. Hence, from **Table 4**, the best method for the glass dataset is XGBoost, as it obtains the highest result for sensitivity (100%), specificity (100%), precision (100%), F1-sore (100%), and g-mean (100%) for the test set.

Based on **Table 5**, AdaBoost, XGBoost, and logistic regression perform well and are stable in predicting positive cases. AdaBoost and XGBoost have similar performances which are a better than logistic regression, as all the performance measures are higher for the boosting methods. However, all three methods face an overfitting issue, since the training set performs better than the testing set for each method. The best method for the ecoli dataset is AdaBoost, as it has the highest sensitivity (training: 84%, test: 80%), specificity (training: 100%, test: 96.67%), precision (training: 100%, test: 72.73%), F1-score (training: 91.3%, test: 76.19%), and g-mean (training: 91.65%, test: 87.93%) for both training and test sets.

Finally, from **Table 6**, XGBoost shows the highest performance for sensitivity and all other performance measures in comparison to AdaBoost and logistic regression. The overall abilities of all three methods are the best when the imbalance ratio is not severe, which is at 25%, and with a large sample size of $n = 2000$.

4. Conclusions

In this study, we have reviewed how imbalanced datasets can be dealt with using the existing approaches, namely boosting algorithm and traditional classification methods by comparing the performance of AdaBoost, XGBoost, and logistic regression. We applied AdaBoost, XGBoost, and logistic regression on nine sets of simulated imbalanced datasets with three different minority percentages and sample sizes. All three methods have an overfitting issue, as their performance on the training set is better than on the test set. The study did not find one best method for the simulated datasets, as each minority percentage has a unique best-performing method. For the simulated imbalanced datasets, logistic regression performed the best for the 5% minority dataset, XGBoost is the best method for the 10% minority dataset, and AdaBoost is the best method for the 25% minority dataset. For the real datasets, XGBoost performed the best for two out of three datasets, namely the ecoli and wifi datasets. The inconsistent results could have been caused by underlying issues of multicollinearity and outliers in the simulated datasets, which have not been examined due to time constraint. Finally, AdaBoost, XGBoost, and logistic regression are the best methods for handling imbalanced datasets when the imbalance is not severe; however, logistic regression is the best method for severe imbalanced datasets compared to boosting algorithm.

Acknowledgements

The authors are grateful for the financial support provided by the Faculty of Computer and Mathematical Sciences Universiti Teknologi MARA, Shah Alam, Selangor.

REFERENCES

[1] A. Sonak and R. A. Patankar, "A Survey on Methods to Handle Imbalance Dataset," *Int. J. Comput. Sci. Mob. Comput.*, 2015.

[2] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *Int. J. Emerg. Technol. Adv. Eng.*, 2012.

[3] B. W. Yap, K. A. Rani, H. A. Abd Rahman, S. Fong, Z. Khairudin, and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Lecture Notes in Electrical Engineering*, 2014, doi: 10.1007/978-981-4585-1 8-7_2.

[4] Y. B. Wah, H. A. A. Rahman, H. He, and A. Bulgiba, "Handling imbalanced dataset using SVM and k-NN approach," in *AIP Conference Proceedings*, 2016, doi: 10.1063/1.4954536.

[5] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*. 2016, doi: 10.1007/s13748-016-0094-0.

[6] G. M. Weiss, "Mining with Rarity: A Unifying Framework," *SIGKDD Explor.*, 2004, doi: 10.1145/1007730.1007734.

[7] S. Fong, R. P. Biuk-Aghai, Y. W. Si, and B. W. Yap, "A lightweight data preprocessing strategy with fast contradiction analysis for incremental classifier learning," *Math. Probl. Eng.*, 2015, doi: 10.1155/2015/125781.

[8] H. A. A. Rahman, Y. B. Wah, H. He, and A. Bulgiba, "Comparisons of ADABOOST, KNN, SVM and logistic regression in classification of imbalanced dataset," in *Communications in Computer and Information Science*, 2015, doi: 10.1007/978-981-287-936-3_6.

[9] J. Song, X. Lu, and X. Wu, "An improved adaboost algorithm for unbalanced classification data," in *6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009*, 2009, doi: 10.1109/FSKD.2009.60 8.

[10] T. SRIVASTAVA, "How to use XGBoost algorithm in R in easy steps," *Analytics Vidhya*, 2016. .

[11] R. E. Schapire, "A brief introduction to boosting," in *IJCAI International Joint Conference on Artificial Intelligence*, 1999.

[12] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud," in *Proceedings - 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018*, 2018, doi: 10.1109/BigComp.2018.00044.

[13] D. Zhang, L. Qian, B. Mao, C. Huang, B. Huang, and Y. Si, "A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost," *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2818678.

[14] S. Ahmad, N. M. Ramli, and H. Midi, "Outlier detection in logistic regression and its application in medical data analysis," in *CHUSER 2012 - 2012 IEEE Colloquium on Humanities, Science and Engineering Research*, 2012, doi: 10.1109/CHUSER.2012.6504365.

[15] H. A. Abd Rahman and B. W. Yap, "Imbalance effects on classification using binary logistic regression," in *Communications in Computer and Information Science*, 2016, doi: 10.1007/978-981-10-2777-2_12.

[16] Shanmugasundaram, N., Sushita, K., Kumar, S.P., Ganesh, E.N. "Genetic algorithm-based road network design for optimising the vehicle travel distance" International Journal of Vehicle Information and Communication Systems, 2019, 4(4), pp. 355–374.

[17] Shanmugasundaram, N., Ganesh, E.N., Kumar, N. "Estimation of power analysis in WLAN infrastructure" International Journal of Engineering and Technology(UAE), 2018, 7(2), pp. 198–200.