Extreme Gradient Boosting using Squared Logistics Loss function

[1]Anju, [2]Akaash Vishal Hazarika

Students
Department of Computer Science & Engineering,
National Institute of Technology Delhi, New Delhi,India

***Abstract*—Tree boosting has empirically proven to be a highly effective approach to predictive modeling. It has shown remarkable results for a vast array of problems. More recently, a tree boosting method known as XGBoost has gained popularity by winning numerous machine learning competitions.**

In this manuscript, we will investigate how XGBoost differs from the more traditional ensemble techniques. Moreover, we will discuss the regularization techniques that these methods offer and the effect these have on the models.

In addition to this, we will attempt to answer the question of why XGBoost seems to win so many competitions. To do this, we will provide some arguments for why tree boosting, and in particular XGBoost, seems to be such a highly effective and versatile approach to predictive modeling. The core argument is that tree boosting can be seen to adaptively determine the local neighborhoods of the model. Tree boosting can thus be seen to take the bias-variance tradeoff into consideration during model fitting. XGBoost further introduces some improvements which allow it to deal with the bias-variance tradeoff even more carefully. We performed these techniques in outliers also.

Additionally, we perform XGBoost with a loss function named squared logistics loss (SqLL) and find out loss percentage. Also we applied this SqLL with other algorithm also.

***IndexTerms*—XGBoost, AdaBoost, Random Forest, Big Data, Boosting, Loss, Logistics Loss Function.**

I. INTRODUCTION

In the course of recent decades, machine learning and data mining have turned out to be one of the backbones of data innovation and with that, a somewhat central, although typically hidden, part of our life. With the constantly expanding amounts of data getting to be noticeably accessible, there is justifiable reason to consider that information mining will turn out to be a significant element for technological advancements. Nowadays, machine learning and information mining have turned into a vital piece of human life. In all aspects of life, applications of these two are utilized. Example applications include fraud detection, e-mail protection and in recommender system it helps to find out the product of user's choice etc [1].

Further, Boosting is the most widely used tool used in machine learning.Kearns and Valiant (1988, 1989)[2] [3] asked a question, "Can weak learners consolidated into a strong one?". Boosting algorithm is a reply of this question. Thisreply is given by Robert Schapire's in a 1990 paper [4]. So Kearns and Valiant has played an important role in the enlargement of boosting [5].It improves the accuracy of prediction of various classification models. Boosting technique is an ensemble technique created by combing various weak learners to build a strong learner with higher precision. Weak learners are those indicators that give more precision than random guessing. However, strong learners are those classifiers that give maximum accuracy and hence coined as the base of machine learning. Boosting technique is employed when the dataset is large and high predictive power is the vital requisite of the application. Further, it is also used to reduce the bias and variance in the prediction models. However, the technique also solves the over-fitting problem for smaller dataset. Additionally, it has wide application area and applies on numerous classification techniques viz. feature selection, feature extraction, and multi-class categorization. The applications of boosting include medical area, text classification, page ranking and business and so on.[6]

Furthermore, Boosting technique is a type of ensemble method, which is used when there is a collection of many weighted same or different type of predictors.

However in this technique, a collection of several hypothesis is selected and eventually their prediction is combined. For example, if 50 decision trees are generated over same or different training data set then a new test dataset is created and voted for best classification [7].

To illustrate, a simple example of boosting is suppose we have to identify about insurance company i.e. whether it is fraud or not and following key points are there:

- Company has a high profit = Strong
- Company has a proper mail id and proper website = Strong
- It gives a proper receipt after paying the amount = Strong
- Large number of customers = Strong
- Behavior with the customers = Weak
- Queries are solved on the toll free number = Weak

In the above query, there are four strong and two weak points and when we collect all the points. Then according to the majority it can be inferred that the company is not fraud and anyone can invest his money on this. Hence, boosting considers assigning weights to various points and then combining the results to predict the class. In addition, several boosting algorithms are already in place. The most widely used are:

1. Adaptive Boosting (AdaBoost)
2. Gradient Boosting
3. Extreme Gradient Boosting (XGBoost)
4. Random Forest

II. Miscellaneous Terms

A. Adaptive Boosting (AdaBoost)

AdaBoost also known as Adaptive Boosting algorithm is proposed by Freund and Schapire[8]. It is an ensemble learning technique where multiple weak learners are consolidated to create one strong learning algorithm. The algorithm starts by selecting a base classification algorithm (e.g. Na¨ıve Bayes) and repetitively enhancing its prediction accuracy by draining the inaccurately classified samples in the training dataset. Initially, AdaBoost assigns same weights to all the training samples and selects a base classifier algorithm. Further, for every iteration, the base algorithm classifies the training samples and the weights of the inaccurate classified samples are increased. The algorithm iterates *n* times, repeatedly applying base classification algorithm on the training dataset with new calculated weights. At the end, the final classification model is the weighted sum of the *n* classifiers [9].

B. Gradient Boosting

Gradient boosting is an effective off-the-shelf strategy for creating accurate mod-els for classification problems. The technique has empirically proven itself to be highly effective for a vast array of classification and regression problems. As stated previously, gradient boosting is a variant of ensemble method, meaning that the prediction is consolidated from several simpler predictors. The aim of this method is to train a collection of decision tress, given the case that the training of single decision tree is known apriori. The technique is called “boosting” in light of the fact that we anticipate that combined results will provide more accuracy than a sin-gle learner. However, in this method the boosting is visualized as an optimization problem, where the objective of the technique is to minimize the loss of the classifier model by adding one weak learner at a time as done in a gradient descent. Gradient boosting is also called stage-wise additive classifier as a new weak learner is added at one time and the previously classified weak learners are left frozen i.e. unchanged for that iteration.

C. Extreme Gradient Boosting (XGBoost)

XGBoost stands for Extreme Gradient Boosting, developed by Tianqi Chen. Since its introduction in 2014, XGBoost has quickly become among the most popular methods used for classification in machine learning. XGBoost is an extension of gradient boosting by Friedman[10]. XGBoost has been successfully used in recent Kaggle competitions, usually as an integral part of the winning ensemble

It implements a variety of Gradient Boosting algorithms, including Generalized Linear Model (GLM) and Gradient Boosted Decision Tree (GBDT). The focus is on scalability. XGBoost differs from Random Forests mainly in the way it creates the tree ensemble. Trees do not have to be trained on a subset of the data or a subset of the features. The ensemble is built sequentially. In each round, k-trees are used to classify examples into k classes. New trees focus on previously misclassified examples to improve the discriminative power of the ensemble. Boosting increases the risk of overfitting, to prevent this, XGBoost employs early stopping. XGBoost can use any loss function that specifies a gradient. It consists linear model. It also contains tree learning method. It is faster than other because of parallel computation. It can be used in regression, classification[11], ranking [12] and in online advertise system[13] etc. Various objective functions are support by XGBoost. In this users can easily define their own objectives. It also has linear model solver algorithm. And also this is a package in R version [14].It is an implementation over the gradient boosting. XGBoost is greedy in nature so it follows greedy approach. It has high performance and speed. XGBoost is a scalable system for learning tree ensemble method. It is used for wide number of applications and it also supports outdoor memory [15]. Due to parallel computation process it is faster than other boosting algorithms [16].The reason behind the higher performance of XGBoost is that it is scalable in nature. Additionally, it has following advantages over other algorithm:

- Due to parallel processing process, it has faster performance than gradient boosting.
- It controls the over fitting problem.
- It gives a better performance result on many datasets.
- Basically it is a tree building algorithm.
- It is used for classification, regression and ranking with custom loss functions.

D. Random Forest

Another ensemble technique that is widely used in Machine Learning is random forest. It is used in classification, regression and many more prediction problems. At training time, multiple decision trees are created and the output is the mean or average prediction of each trees. The algorithm is proposed by Tin Kam Ho [17]. Random forest follows following steps:

- Using the bagging process sampling of training dataset takes place. It gives a number of trees.
- Nodes are split according to some splitting criteria.

- Due to splitting criteria, data is divided into each node.
- Classification takes place on leaf node.
- After trained for trees, test data is sampled. Each sample is given to all trees.
- At the leaf node classification takes place.
- At last, the class of the test dataset is decided by majority voting or average process.

Furthermore, in random forest algorithm, the classifier shows low bias and high variance. Random Forest follows the parallel computation process. The algorithm that is used for training and testing process is bootstrapping. However, for very iteration, data is split into number of trees using bagging process. Bagging process divides the whole dataset and creates samples. Then, classification is done on these samples using decision trees. Further the classifier predicts the classes of samples and final class is predicted by the majority voting or it can be the simple average. In various situations, Random Forest gives more exact predictions when distinguished with simple Classification and Regression Tree (CART) models or regression models.

E. Loss Functions in Boosting Algorithms

Boosting is a representation on gradient decent algorithm for loss functions[18] [19].

Suppose that $(u_1, v_1), ..., (u_n, v_n)$ is observed, where u_i is the value of input space χ and v_i is the class label takes 1 or -1 value. The collection of weak learners is denoted by $WL = (w_i(u) : \chi \rightarrow (1, -1)|i = (1, ..., I))$,where each learner gives class label for the input [20].Strong learner SL is made by consolidated weak learners, where SL is given by,

$$Sl = \sum_{i=1}^{l} (\alpha_i\, w_i(\mu))$$

Loss functions are generally used for classification problems. The loss given by SL over sample (u, v) is $l(-vSL(u))$, where $l : R \rightarrow R$ is continuous and differentiable function except finite points. It is also called growing functions, so loss function is given by

$$L(Sl) = (1/n) \sum_{i=1}^{n} l(-v_i * Sl(u_i))$$

So, in simple term, a loss function is the cost of the error between the prediction $z(b)$ and the observation a at the point b. Loss function is convex function. A convexfunction shows that there are no local minima. In every optimization problem, the main task is to minimize the loss or cost function. It may be its objective function also. And to minimize the conditional risk, a loss function is derived that is used for outliers.

Typically, a loss function is a value which is based on predicted and true value. And it is the difference between actual and predicted value. Loss function is the penalty for misclassified data points in any classification problem. It is used to analysis the performance of linear regression.

Furthermore, it classifies the data points with high accuracy and it is fits the outliers. Loss function always affects the accuracy. Loss function is convex in nature. Loss function gives two type of error one is positive part error and second is negative part error. Negative part error always decreases the accuracy. Positive part truncation makes strong to any boosting algorithm.

In binary classification, loss function shows the minimum probability error but from the computational point it is difficult to tackle. Savageboost is faster than other and it is more robust on outliers than others. Moreover it has a faster convergence rate [21].

Many type of loss function are available for different task for example hinge loss is used in SVM, exponential loss is used in AdaBoost .It is valid only for global minimization. The loss function which gives local optimization is still pending to be find out. In addition, It is based on greedy approach it would stop adding weak classifier when the error become minimum. A boosting algorithm named direct boost is given for binary classification named direct boost. Ada is work sequential. Error that is given by ensemble classifier is confined by distribution of margin. In this relaxation term is used .Relaxation is used to overcome the problem when it is impossible to maximize the margin along coordinate direction. After setting the positive and negative relaxation algorithm allow this on a single coordinate to change margin [22].

III. Experiment and Performance Evolution

A. Dataset

In the manuscript, we have taken the soyabean crops (disease) data set which can be easily available on the link https://archive.ics.uci.edu/ml/machine-learning-databases/soybean. In this link, two data sets are available one is small and other is the large. Small soyabean dataset contains small data.

Since, the dataset holds 19 classes. It contains 35 categorical attributes, nominal and ordered value also. The values for attributes are fixed numerically, with the first value encoded as ”0,” the second as ”1,” and so forth. Does not apply values are shortly abbreviated as "dna". "?" represent the unknown value. It is used for classification purpose. The simulation performed on 64 bit operating system with 4 gb ram and i5 processor.

B. Experiment

Enhancing the prediction power of the boosting algorithm used in several classification problems. The boosting algorithm considered in this work is state of art extreme gradient boosting (XGBoost) algorithm which is proven to be more efficient than the other counterparts as it can proven parallel computation at the same time. However, the confidence of classification points achieved from the XGBoost can further be enhanced using squared logistics loss (SqLL) function. The main focus of this work is to present an efficient boosting algorithm in terms of prediction power, which reduces the loss function of the target variables as well. SqLL is given by :

$$l(\phi) = \sum_j [\ (\alpha_j log_{10}(1 + e^{-\alpha_j}))^2 + ((1 - \alpha_j)log_{10}(1 + e^{-\alpha_j}))^2]$$

Where,

$$l(\phi) = loss\ Function$$

$$\alpha_j = Target\ Value$$

Moreover, in this we have given a detailed analysis of XGBoost and compare it with other techniques on the basis of accuracy. Further we analyze the result in terms of loss when we apply SqLL on the errors.

C. Tools and Platform

R is an open source distribution system. It is statics software which is easy to use. It is also used for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Packages are easily available in this.

D. Implementations

It is previously explained that XGBoost has fast performance and speed due to parallel computation. It deals with the categorical as well as numerical dataset. To take the advantage of this technique we applied it on soyabean dataset the result obtained minimum error means maximum accuracy with high speed. In this section, a comparison is made between three ensemble methods (XGBoost, Random Forest, Adaboost) on the basis of accuracy.

E. Comparison

The comparison between these techniques on the basis of accuracy is given in Table 1

Techniques	**Total Error (%)**
AdaBoost	0.8957655
Random Forest	2.125
XGBoost	0.081433

Table 1Comparison between techniques

F. Error Graph

The error graph of random forest is given in figure 1, adaboost error graph is given in figure 2 and xgboost error graph is given in figure 3.In these graph,we have noticed that error will be decrease when number of iterations increase.

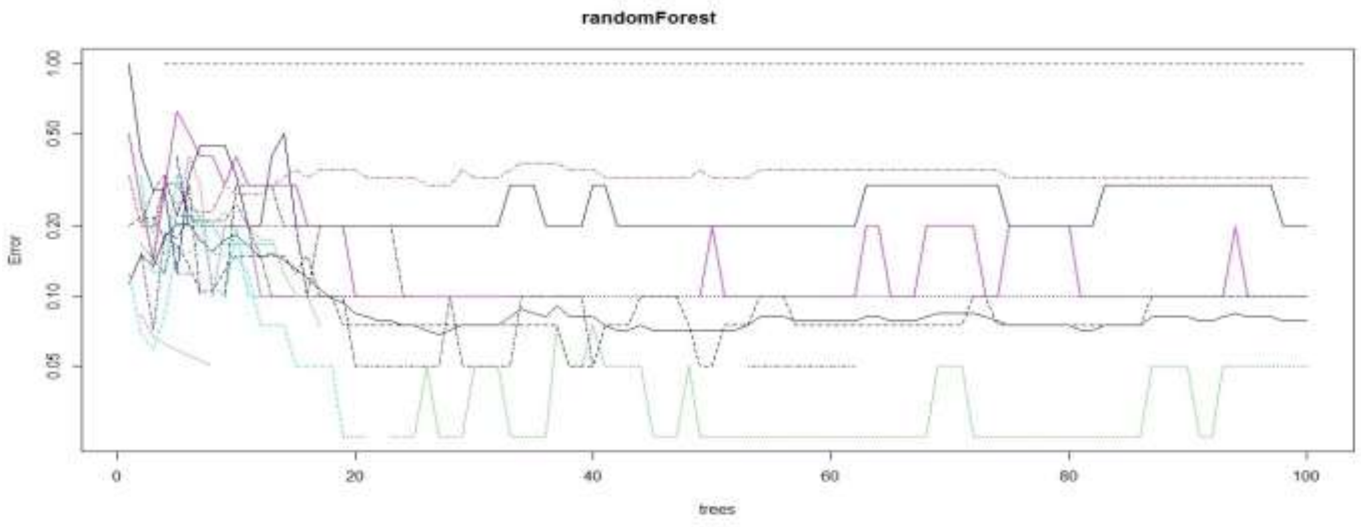

Figure 1 Error graph in Random Forest

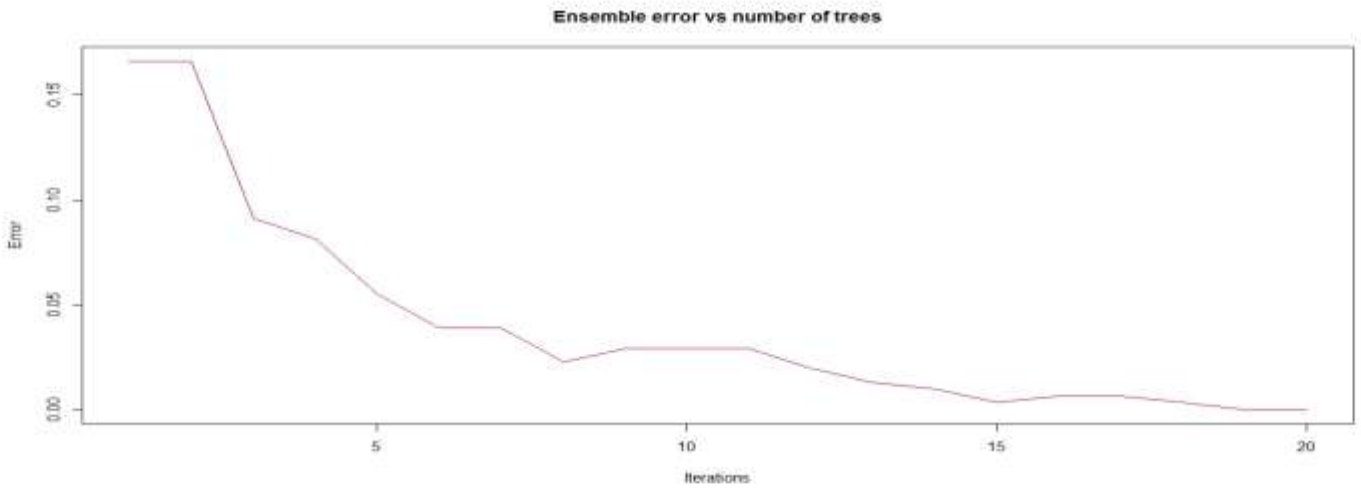

Figure 2Ensemble error vs number of trees in AdaBoost

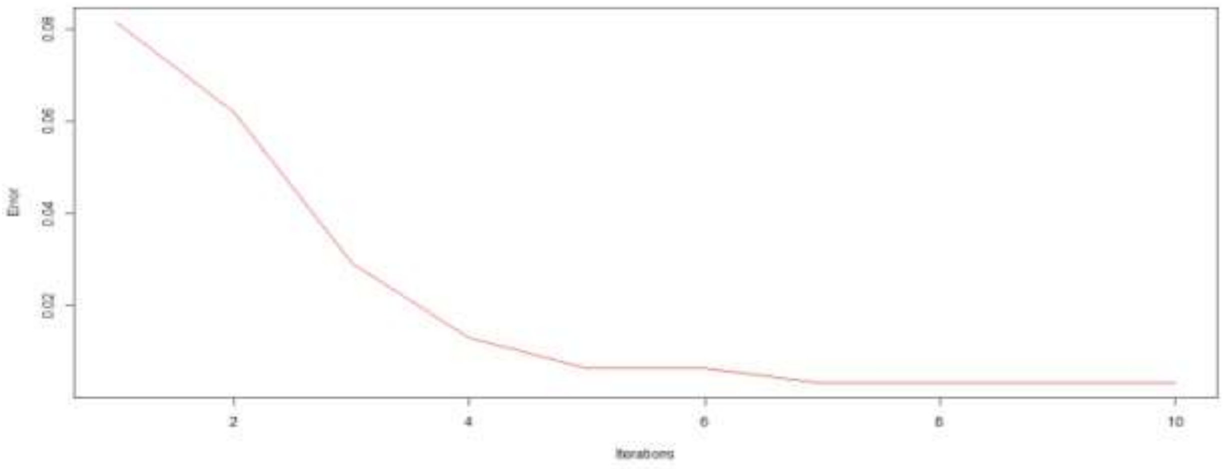

Figure 3XGBoost Error graph vs number of iterations

G. Comparison between Techniques

The comparison between these techniques on the basis of accuracy is given in Table 2

Techniques	**Total Error (%)**
AdaBoost	0.8957655
Random Forest	2.125
XGBoost	0.081433

Table 2Comparison between techniques

*H. **Gross Error Sensitivity (GES)***

When the data is changed slightly it is called outliers. GES is a matrix where we deliberately add outliers in the dataset and use it to compare in table 4.5 at different portion of outliers. For example: Age=15—learn—class=student, and if we change age from 15 to75 then in which class it falls. If it shows retired then it corrects predictor otherwise wrong. In our dataset we make a comparison between different techniques after adding 0%, 2% and 4% outliers,given in Table 3.

Outliers	**0%**	**2%**	**4%**
AdaBoost	0.8957655	0.9283388	1.068404
RandomForest	2.125	7.17	7.82
XGBoost	0.081433	0.08469	0.08469

Table 3GES (Gross Error Sensitivity)

I. Techniques comparison with squared logistic loss

When we apply the squared logistic loss function on different techniques, then the result are very encouraging depicting that error in XGBoost has minimum but after applying squared logistics Random forest gave us minimum loss. Result of this can be easily seen in the Table 4.

Techniques	**Total Loss**
Random Forest with SqLL	1.577643
AdaBoost with SqLL	1.764983
XGBoost with SqLL	1.717639

Table 4Comparison among techniques with SqLL

IV. CONCLUSION

To conclude, we can say that boosting algorithm is very vast in itself and also it has many interpretations. AdaBoost is better than a random imagination and also we saw that XGBoost has a fast performance due to parallel computation while other boosting algorithm works on serial computations. Missing values is handled in these algorithms. Over fitting problem is also overcome by these algorithms. In boosting system, various weak learners are consolidated and give a strong learner with higher accuracy. Therefore, bias and variance are considered important parameters to measure the accuracy of these algorithms. The better algorithm is the one which provides high bias and low variance. Both XGBoost and AdaBoost depict the same. But Random forest shows the opposite. Accuracy is also impacted by the cross validation of error. All the three algorithms implement the cross validation of error and hence are more accurate than single learner. To state the comparative analysis of the accuracy of all the three algorithms, accuracy of XGBoost is maximum and random forest shows the least accuracy amongst all. Over fitting of data occurs due to the branches involving noisy data or outliers. It is imperative to reduce the overfitting problem to enhance the accuracy of the learners. Pruning is done to remove the overfitting of data. The pruning can be done in two ways: Pre pruning and post pruning. Prepruning involves the avoidance of overfitting problem while post pruning removes the overfitted data after the learner is created. XGBoost and Random Forest avoids overfitting problem, Adaboost does not avoid the problem completely but it is less prone to overfitting.

Moreover, a lot of analysis was performed on the data to identify patterns and outliers which would booster delay the prediction algorithm. Data mining methods like AdaBoost, Random Forest Regression and XGBoost were implemented and their results compared. XGBoost which is an improved gradient boosting algorithm was observed to perform the best at prediction.

Further, when we apply SqLL on these algorithms then random forest gives better result than the Xgboost and adaboost. It shows minimum loss.

In summary, XGBoost which is implemented on gradient boosting algorithm and follows the greedy approach is best performing boosting algorithm in terms of computation performance and accuracy. In short, performance of Xgboost is better than other boosting algorithm. And after applying SqLL Random forest shows best result other than two.

V. Future Work

In this manuscript, we proposed the implementation of SqLL in XGBoost to enhance the predictions power of boosting used in classification. The proposed techniques were implemented on a single data set i.esoyabean crop dataset. Future scope includes the stimulation of proposed methodology in a broader range of dataset.

Further, in our future scope we would try to devise few techniques to determine whether a particular dataset is conducive to XGBoost. Furthermore, the proposed SqLL function is stimulated only for XGBoost. So further work is to apply the loss function on other techniques to see it indeed works for other boosting algorithm or not.

References

[1] T. Chen and C. Guestrin, "Xgboost: Reliable large-scale tree boosting system," in Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2015, pp. 13–17.

[2] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean for-mulae and finite automata," Journal of the ACM (JACM), vol. 41, no. 1, pp. 67–95, 1994.

[3] M. Kearns, "Thoughts on hypothesis boosting," *Unpublished manuscript*, vol. 45, p. 105, 1988.

[4] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[5] L. Breiman*et al.*, "Arcing classifier (with discussion and a rejoinder by the author)," *The annals of statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[6] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p.1612, 1999.

[7] Z.-H. Zhou, Ensemble methods: foundations and algorithms. CRC press, 2012.

[8] M. Culp, K. Johnson, and G. Michailidis, "ada: An r package for stochastic boosting," *Journal of Statistical Software*, vol. 17, no. 2, p. 9, 2006.

[9] R. E. Schapire, "Explaining adaboost," in *Empirical inference*. Springer, 2013 ,pp. 37–52.

[10] J. H. Friedman, "Greedy function approximation: a gradient boosting ma-chine," *Annals of statistics*, pp. 1189–1232, 2001.

[11] P. Li, "Robust logitboost and adaptive base class (abc) logitboost," *arXivpreprint arXiv:1203.3491*, 2012.

[12] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.

[13] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for OnlineAdvertising*. ACM, 2014, pp. 1–9.

[14] T. Chen and T. He, "Package 'xgboost'- r package version 0.3-0," in *TechnicalReport*, 2015.

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 785–794.

[16] T. Chen and T. He, "Xgboost: extreme gradient boosting," *R package version0.4-2*, 2014.

[17] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8,pp. 832–844, 1998.

[18] J. Friedman, T. Hastie, R. Tibshirani*et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[19] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Advances in neural information processing systems*, 2000, pp. 512–518.

[20] T. Kanamori, T. Takenouchi, S. Eguchi, and N. Murata, "The most robust loss function for boosting," in *Neural Information Processing*. Springer, 2004, pp. 496–501.

[21] H. Masnadi-Shirazi and N. Vasconcelos, “On the design of loss functions for classification: theory, robustness to outliers, and savageboost,” in *Advances inneural information processing systems*, 2009, pp. 1049–1056.

[22] S. Zhai, T. Xia, M. Tan, and S. Wang, “Direct 0-1 loss minimization and margin maximization with boosting,” in *Advances in Neural Information ProcessingSystems*, 2013, pp. 872–880.