Int. J. Advance Soft Compu. Appl, Vol. 10, No. 2, July 2018
ISSN 2074-8523

The Accuracy of XGBoost for Insurance Claim Prediction

Muhammad Arief Fauzan[1], Hendri Murfi[1]

[1]Department of Mathematics, Universitas Indonesia
muhammad.arief52@sci.ui.ac.id, hendri@ui.ac.id

Abstract

The increasing trend of claim frequency and claim severity for auto-insurance result in need of methods to quickly file claims while maintaining accuracy. One of them is machine learning that treats the problem as supervised learning. The volume of the historical claim data is usually large. Moreover, there are many missing values for many features of the data. Therefore, we need machine learning models that can handle both data characteristics. XGBoost is a new ensemble learning that should be very suitable for both data characteristics. In this paper, we apply and analyze the accuracy of XGBoost for the problem of claim prediction. We also compare the performance of XGBoost with that of another ensemble learning, i.e., AdaBoost, Stochastic GB, Random Forest, and online learning-based method, i.e., Neural Network. Our simulations show that XGBoost gives better accuracies in term of normalized Gini than other methods.

Keywords *claim prediction, large volume, missing values, ensemble learning, XGBoost*

1 Introduction

Auto-insurance claim is a request for financial coverage caused by automobile-related loss or sustained damage from a policyholder. An auto-insurance provides coverage of bodily injury, property damage, personal injury, comprehensive physical damage, and collision. Traditionally, the process of filing an auto insurance claim, as well as the rationale to accept one, is manually handled and

can be very different depending on the cause of damage, profile of the policyholder, and many other factors[1].

According to an Insurance Information Institute study, claim frequency - number of claims per vehicle - as well as claim severity - average size of the claim - is in an increasing trend, e.g., property damage claim severity and frequency increased by 11.5% and 2.9% respectively for US auto-insurances in Q1 2014 – Q1 2016[2]. Furthermore, US average expenditures for auto-insurance from 2009 to 2015 is also in an increasing trend, reaching USD 889.01 by 2015 from USD 786.65 in 2009 [1]. These result in need of a faster, reliable method to file auto-insurance claims to be able to keep up with the increasing trends of claim severity and frequency.

Claim prediction is an important process in the insurance industry because the insurance company can prepare the right type of insurance policy for each potential policyholder. Inaccuracies in the prediction of vehicle insurance claims will raise the price of the insurance policy for the good driver and reduce the price of the policy for the driver who is not good. More accurate prediction capability allows the insurance industry to adjust pricing better and makes car insurance coverage more accessible to more drivers.

From machine learning point of view, the problem of claim prediction can be categorized as supervised learning [2, 3]. Given the historical claim data, we need to build a machine learning model that predict if a driver will initiate an auto insurance claim. The volume of the historical data is usually large. Moreover, there are many missing values for many features of the data. Therefore, we need machine learning models that can handle both data characteristics. There are some machine learning paradigms relevant in the big data, especially volume context. They include ensemble learning and online learning [4]. Ensemble learning combines multiple models of data chunk to obtain better accuracies than those obtained from any constituent model. Online learning uses data streams for training, and model can learn one instance at a time.

XGBoost is a new ensemble learning for classification problem. It is a novel gradient tree-boosting algorithm that offers efficient out-of-core learning and sparsity awareness [5]. The out-of-core learning is a set of algorithms processing data that cannot fit into the memory of a single computer, but that can easily fit into some data storage such as a local hard disk. Therefore, XGBoost is a supervised learning algorithm that should be very suitable for the problem of claim prediction.

In this paper, we apply and analyze the accuracy of XGBoost for the problem of claim prediction. We also compare the performance of XGBoost with that of another ensemble learning, i.e., AdaBoost, Stochastic GB, Random Forest, and

[1] https://www.claimsjournal.com/news/national/2013/11/21/240353.htm
[2] https://www.iii.org/fact-statistic/facts-statistics-auto-insurance

online learning-based method, i.e., Neural Network [6]. Our simulations show that XGBoost gives better accuracies in term of normalized Gini than other methods.

The rest of the paper is organized as follows: In Section 2, the reviews of related works are presented. Section 3 describes the problem formulation or methodology. In Section 4, we discuss the results of simulations. Finally, we give the conclusion in Section 5.

2 Related Work

There have been several papers that tackled the problem of claim prediction using machine learning. Weerasinghe et al. compared which machine learning method performs best in predicting the claim size of a policyholder. They compared three machine learning method, i.e., neural networks, decision tree, and multinomial logistic regression. Their results indicated that neural networks were the best predictor [2]. To predict whether a policyholder files a claim or not, Smith et al. examined some machine learning methods such as decision tree and neural networks and discussed the impact of the case study on the insurance company [3].

The volume of the historical claim data is usually large. Moreover, there are many missing values for many features of the historical claim data. The above previous works did not consider both big volume and missing value issues in their works. Therefore, we focus on examining the machine learning methods that are the most suitable method for the problem of claim prediction with big training data and many missing values.

2.1 Missing Values

Datasets obtained from real-world resources such as claim data often contain missing values, intentional or not. Other examples are medical surveys and ad-click data. Medical surveys may omit some fields that are deemed irrelevant to the respondent's current condition, and an ad-click may fail to register into an ad-click report because of blocked network traffic.

Some works propose imputation approaches for missing values on certain datasets. In [7], the authors propose an imputation measure for missing categorical values in a medical dataset, using concepts of similarity between a row whose missing values are removed and a subset of the dataset that contains no missing values, whose corresponding columns are removed. In [8], a K-means clustering method to impute missing data is proposed. First, the method creates clusters of complete instances. The method then iteratively imputes the missing values of instances using K-nearest neighbor method starting from instances with the least missing values to instances with the most missing values and updates the centroids of the clusters. A similar method of imputation is proposed in [9], where clusters of the whole dataset are made using K-means clustering method. Then, instances belonging in a cluster that contains missing values are imputed using a

kernel-based method taking values from instances in the same cluster that do not contain missing values.

2.2 Big Data

The large volume of data to consider in predicting whether a policyholder will claim or not has made the problem of claim prediction turns into big data paradigms. Regarding handle the large volume of data, some paradigms are usually used by machine learning, e.g., online learning and ensemble learning [4].

Online or incremental learning uses data streams for training, and model can learn one instance at a time [10]. This approach enables the processing of the large volume of data. A popular example of online learning based machine learning is neural networks.

Ensemble learning breaks the large volume of claim data into small ones, training models on a small subset, and combining results with high accuracy [11]. Decision tree is an example of averaging-based ensemble learning [12], while AdaBoost and stochastic gradient boosting are examples of boosting-based ensemble learning [6].

2.3 XGBoost

Recently, a new boosting-based ensemble learning called XGBoost is proposed [5]. It is a novel gradient tree-boosting algorithm that offers efficient out-of-core learning and sparsity awareness. Therefore, XGBoost is a supervised learning algorithm that should be very suitable for the problem of claim prediction with big training data and missing values. The common previously used methods such as random forest and neural network still can not handle missing values. The methods need other mechanisms to handle the missing values.

The robustness of XGBoost results in increasing usages of the method in many other applications. As an example, Aler er al. utilizes XGBoost in the field of direct-diffuse solar radiation separation by creating two models [13]. This first model is an indirect model that uses XGBoost as a level-1 learner to learn from the result of conventional solar radiation separation models obtained from various literature sources fitted into a dataset. The second model is a direct model that directly fits XGboost into a dataset. Another example is in [14] that uses XGBoost to recommend items to a user in a recommender system, by using features extracted from a user-item pair using intricate feature engineering. In this paper, we examine XGBoost as a predictor for the problem of claim prediction.

3 Claim Prediction as a Machine Learning Problem

The goal of claim prediction is to predict the probability whether a policyholder files a claim or not given some information about policyholders. Let X be a collection of instances $\boldsymbol{x_i}$ that represent all the known information of the i-th policyholder, and $Y = \{0,1\}$ be the possible output class, with 0, 1 being

categorical values that represent 'policyholder does not file claim' and 'policyholder files claim' respectively. Then, claim prediction seeks the probability pair

$$(\Pr(Y = 0 \mid X = \boldsymbol{x_i}), \Pr(Y = 1 \mid X = \boldsymbol{x_i}))$$

where each probability is provided by a probability classifier, after being fitted to X. In practice, we are only concerned on finding the second entry of said pair, which is the probability of the i-th policyholder filing a claim. Note that this does not result in any loss of generality, since there are only 2 possible output classes.

3.1 Normalized Gini Coefficient as a Model Evaluation Metric

Fitting a model to the dataset $\{(\boldsymbol{x_i}, y_i) |\ 1 \leq i \leq n\}$ creates a conditional predicted probability$f(\boldsymbol{x})$, where $f(\boldsymbol{x_i}) = \Pr(Y = 1 \mid X = \boldsymbol{x_i})$. To evaluate the model, this paper will use normalized Gini coefficient as a model evaluation metric. Gini coefficient is originally used to calculate a nation's inequality of income distribution [3], but it was soon adapted into a model evaluation metric for classification problems, as it was found to be closely related to Receiving Operating Characteristic, another often-used metric for classification problems [15]. One of the possible methods to calculate normalized Gini coefficient for a binary classification problem where the classes are defined as '0's and '1's, of which claim prediction falls into, is described below.

Given a target variable vector $\boldsymbol{y} = [y_1, y_2, \dots, y_n]^T$where $y_i \in \{0,1\},\ 1 \leq i \leq n$ is the correct class of the i-th observation and a predicted probability vector $\boldsymbol{f}(\boldsymbol{x}) = [f(\boldsymbol{x_1}), f(\boldsymbol{x_2}), \dots, f(\boldsymbol{x_n})]^T$ where $f(\boldsymbol{x_i}), 1 \leq i \leq n$ is the predicted probability that the i-th instance will fall into the class '1' obtained from a classification algorithm (e.g., logistic regression) fitted to a dataset $\{(\boldsymbol{x_i}, y_i) |\ 1 \leq i \leq n\}$, we first sort $\boldsymbol{f(x)}$ in a descending order, then uses said sorting rule to shuffle $\boldsymbol{y}$ into vector $\boldsymbol{y'} = [y_1', y_2', \dots, y_n']^T$. If X is the amount of iteration needed to bubble-sort $\boldsymbol{y'}$ into $\boldsymbol{y}$, cl_i is the amount of observations in $\boldsymbol{y}$ that are of class 'i', then

$$Y = \frac{cl_0 * cl_1}{2}$$

$$nG = \frac{Y - X}{Y}$$

where nG is the normalized Gini coefficient for the said fitted model[4]. Note that if the algorithm is a random guess, then $X = Y$, which would result in $nG = 0$, and if the algorithm gives a 100% correct prediction for each $\boldsymbol{x_i}$, then $X = 0$, which would result in $nG = 1$.

3.2 Characterization of Missing Values

A real-life dataset for certain domains often contain missing values, and this is no different for claim prediction. Missing values in this domain may be caused by optional fields in the insurance policy that the policyholder may skip, mandatory

[3] http://hdr.undp.org/en/content/income-gini-coefficient.

[4] https://www.kaggle.com/batzner/gini-coefficient-an-intuitive-explanation

fields that the policyholder forgot to fill before the dataset is distributed, removal of noise from the dataset as an early form of data preprocessing. Based on the mechanism of the missing values, there are three types of missing values: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) [16]. While one can remove the observations that have missing values in them, this approach becomes less appropriate as the number of observations with missing values increases since there are fewer observations for the model to train with. There have been studies regarding the various approaches to imputing missing values in datasets, whether by statistical methods [17, 18] or by machine learning methods [19, 20, 21], but this ignores the possibility that the missing value was on purpose as well as introducing possible bias to the dataset. In response, some machine learning methods that can learn despite missing values in the dataset have been developed [5, 13, 14]. This paper will use methods developed at [5].

4 XGBoost

XGBoost is a novel gradient tree boosting method introduced by Chen and Guestrin [5]. It first applies a set of Classification and Regression Trees (also known as CART) as weak learners and then boosts the performance of the trees by creating an ensemble of trees that minimize a regularized objective function. The algorithm introduced concepts such as sparsity-aware split finding in each tree, cache-friendly approximate algorithms to determine splitting points, and efficient out-of-core calculation to the methods of gradient tree boosting to create an algorithm with very fast computational speed while maintaining good prediction ability.

Given a dataset $\{\boldsymbol{D}, \boldsymbol{y}\}$ and p CARTs $f(\boldsymbol{x})$ as weak learners, the ensemble $F_0(\boldsymbol{x})$ first includes a weak learner $f_0(\boldsymbol{x})$ that learns from the original dataset. Then, the ensemble sequentially adds weak learners that learn from the residual of the previous ensemble. If $t > 0, t \in N$ is the t-th boosting round, then the ensemble $F_t(\boldsymbol{x})$ at the t-th boosting round is

$$F_t(\boldsymbol{x}) = \sum_{i=0}^{t} f_i(\boldsymbol{x}) \tag{1}$$

where $f_t(\boldsymbol{x})$ learns from the residuals of $F_{t-1}(\boldsymbol{x})$, and is the learner that greedily minimizes an objective function L^t, where

$$\begin{aligned} L^t &= \sum_{i=1}^{n} l(y_i, F_{t-1}(\boldsymbol{x_i}) + f_t(\boldsymbol{x_i})) + \Omega(f_t) \\ \Omega(f_t) &= \gamma T + \frac{\lambda \|w\|^2}{2} \end{aligned} \tag{2}$$

where l is a differentiable complex loss function between the i-th outcome y_i and the $(t-1)$-th ensemble's predicted i-th outcome $F_{t-1}(\boldsymbol{x_i})$, and $\Omega(f_t)$ is a function that penalizes tree complexity, with T, w as the amount of leaves and sum of all leaf weights respectively, and γ, λ respectively are the regularization and minimum loss hyperparameters of XGBoost.

As with gradient tree boosting machine learning algorithms, XGBoost can (locally, in a given ensemble) calculate the importance of variables in a dataset [15, 24]. Given a variable V in a CART, the improvement I(V) of a variable that splits a parent node P into child nodes L, R, of which q is the fraction of paths that pass-through L is defined by

$$I(V) = E(P) - (qE(L) - (1-q)E(R)) \tag{3}$$

where $E(K)$ is the weighted squared errors of node K. Importance of a variable in an ensemble is defined as the average of improvement of said variable of all trees in the ensemble.

One of the novelties introduced by XGBoost is the ability to set a default direction in each node of its CART in the form of a split finding algorithm. Given the observations in a node, the algorithm first collects all observations whose value for a feature is not missing in a set I, then calculate the gain obtained from splitting to the left and right for every observations in I. It then stores the maximum split direction and the gain from splitting in said direction from all observations in I as the optimal values for the feature. The process is repeated for every feature, and the default direction is obtained by taking the direction that gives the optimal value from all features. Every observation whose value for the feature that is used for the current node's splitting rule will go to the default direction.

5 Experiments and Results

5.1 Dataset

To build and evaluate the claim predictor, we use publicly available datasets from Porto Seguro through Kaggle[5]. The training data is used to build a model as a predictor of probabilities a person will file a claim next year. Using the testing data, we estimate the accuracy of the model. Some elementary data information from these datasets are:

- There are 595212 observations in the training dataset and 892816 observations in the testing dataset
- There are 57 features and a binary label that is '0' for 'does not file a claim' and '1' for 'files claim'

[5] https://www.kaggle.com/c/porto-seguro-safe- driver-prediction

- Out of all values in the label, only 6% are ‘1’s. This means that the resulting model will be very prone of overfitting.
- Feature titles follow the format ‘ps_(group)_(n-th feature from group)_(none/bin/cat)’, where the groups are ‘ind’ (individual) features, ‘car’ (car) features, ‘reg’ (region) features, and ‘calc’ (calculated values) features. Features that are binary or categorical are given the tag _bin or _cat after the feature number. For example, ps_ind_03_cat is the third feature from the individual feature group, whose entries are of categorical values. The semi-black box nature of the features
- There are massive amounts of NaN values in the dataset, totaling at 846458 NaN values for training dataset and 1270295 NaN values for the testing dataset. Some features, such as ‘ps_ind_03_cat’ has 1028142 NaN values in training and testing dataset combined. Fortunately, XGBoost can handle missing values, preventing the need to impute missing values, which would introduce noticeable bias in the data considering the amount of NaNs

5.2 Feature selection

The semi-black box nature of the dataset (where the exact meaning of each feature is not known, but some characteristics of the features are known) made feature engineering as well as intuitive feature selection hard to approach, and as such feature selection has to be done by more traditional means. In exploring the dataset, the following charts will be used:

- A barplot for all integer-valued features between values and ‘does not claim’ percentage
- A 10-bin histogram for all float valued features between values and ‘does not claim’ percentage
- A heatmap of pairwise correlation between features
- A barplot of correlation between features and target.

Feature selection is made based on the insights gained from the charts above, as well as the pairwise correlations between independent features and the target column and the average of absolute pairwise correlation values between a feature and all other features of the dataset. In particular, features that show little correlation to the target column and low pairwise correlation to other features will not be considered into the model.

The feature selection process filters 24 features from the original 57 features to be further used for the model. In detail,

- All features (20 in total) in the calc group either have low feature-target correlation or low pairwise correlation to other features, including other features on the same calc group.
- The other four features not included in the model (ps_ind_10/11/13_bin, ps_car_10_cat) have low feature-target correlation and low pairwise correlation to other features, thus making it a candidate for deletion.

5.3 Model Fitting, Hyperparameter Tuning, and Results

A preliminary XGBoost classifier is trained to the post-feature selection dataset, then is subject to a 6-stage grid search scheme, each grid search scheme utilizing five cross-validation folds, where each stage optimizes different hyperparameter(s). In detail,

- 1st grid search stage optimizes p, the amount of CART trees contained in the XGBoost ensemble model
- 2nd grid search stage optimizes max_depth, the maximum depth of a single CART tree, and min_child_weight, the minimum sum of instance weight in a tree-child required to induce a split
- 3rd grid search stage optimizes subsample and colsample_bytree, the subsampling ratio of the dataset required to grow CART trees and the subsampling ratio of columns in said subsampled dataset used to construct each tree respectively
- 4th grid search stage optimizes α and λ, the L1 and L2 regularization term on weights of the new features obtained from every boosting round.
- 5th grid search stage optimizes scale_pos_weight, a parameter used to scale the weights of positive observations to pick the CART trees that correctly predicts positive observations, and learning_rate, the ratio of which feature weights obtained after each boosting rounds are scaled by to shorten boosting round times.
- 6th grid search stage optimizes γ, the minimum loss reduction required for a leaf in a CART tree to split.
- (0th stage is the preliminary XGBoost model)

At each stage, the classifier is updated using the result of that stage's grid search. A custom evaluation metric to calculate normalized Gini coefficient is passed as an evaluation metric for the training process of every classifier. All classifiers share some hyperparameters, namely objective ('binary:logistic'; classifies every sample into either '0's or '1's), seed (27; the random number generator seed), and n_thread (6; the amount of CPU cores used in the learning process), and end up having the same learning_rate (0.1). The full list of the hyperparameters of each classifier is given in Table 1 and Table 2.

Five-fold cross-validation is done to get the normalized Gini coefficient of the training dataset, as well as to provide prediction probabilities of the testing dataset to be uploaded to Kaggle for evaluation purposes, which returns normalized Gini coefficients of a fixed 70% and 30% partitions of the testing dataset, set by Kaggle. The normalized Gini coefficients of the training and both partitions of the testing dataset are given in Table 3.

Table 1: Hyperparameters of each XGBoost classifier

Classifier	min_child_weight	p	max_depth	γ	subsample
xgb0	1	500	5	0	0.8
xgb1	1	119	5	1	0.8

xgb2	6	119	4	1	0.8
xgb3	6	119	4	1	0.75
xgb4	6	119	4	1	0.75
xgb5	6	119	4	1	0.75
xgb6	6	119	4	1	0.75

Table 2: Hyperparameters of each XGBoost classifier (cont.)

Classifier	colsample_bytree	scale	α	λ
xgb0	0.8	1	0	1
xgb1	0.8	1	2	8
xgb2	0.8	1	2	8
xgb3	0.65	1	2	8
xgb4	0.65	1	1.1	6
xgb5	0.65	1.9	1.1	6
xgb6	0.65	1.9	1.1	6

Table 3: Normalized Gini coefficient results of each classifier

Classifier	Train	Test (70%)	Test (30%)
xgb0	0.27935	0.26250	0.25124
xgb1	0.27963	0.27932	0.27005
xgb2	0.27981	0.28434	0.27860
xgb3	0.27993	0.28422	0.27993
xgb4	0.28023	0.28455	0.27981
xgb5	0.28113	0.28518	0.28043
xgb6	0.28113	0.28518	0.28043

For comparison purposes, we use two boosting-based ensemble learnings, i.e., AdaBoost and Stochastic GB, an averaging-based ensemble learning, i.e., Random Forest, and online learning, i.e., Neural Network. Since these methods cannot handle missing values, missing values in both training and testing dataset will be filled using the following imputing strategy:

- Missing values in features with binary values will be filled by the value of its median
- Missing values in features with integer values will be filled by the rounded value of its mean
- Missing values in features with float values features will be filled by the value of its mean

The model selection of all comparison models uses a similar grid-search scheme with XGBoost, suited accordingly for the hyperparameter optimization of each model. Table 4 shows the best results of XGBoost and other comparison methods.

Table 4 shows that XGBoost outperforms other methods on both testing datasets. From Table 4, we also see that neural network give better accuracies than random forest which consists of some decision trees. These results relate to the

previous comparison in [2] that showed that neural network gave better accuracies than decision tree.

The results in Table 4 also show that Stochastic GB gives similar scores with XGBoost on the training dataset. Since Stochastic GB has the same underlying methods with XGBoost and Stochastic GB learns from the imputed data whereas XGBoost learns from the unimputed data, this may imply that the imputing strategy works well, at least for the training dataset. AdaBoost is also had a similar underlying method with XGBoost. However, the differences between their methods of creating weak learners may be the reason why their Gini coefficients differ quite much. The weak learners of XGBoost learn from the residuals of the previous ensemble of weak learners, whereas weak learners of AdaBoost learn from the weighted dataset, whose weights are updated from the previous ensemble of weak learners [23].

Table 4: Normalized Gini coefficient results of each method

Classifier	Train	Test (70%)	Test (30%)
XGBoost	0.28113	0.28518	0.28043
AdaBoost	0.2687	0.27271	0.26896
Stochastic GB	0.28139	0.28499	0.27977
Random Forest	0.25841	0.25933	0.25402
Neural Network	0.28023	0.28455	0.27981

6 Conclusion

Claim prediction is an important process in the insurance industry. The volume of the historical claim data is usually large. Moreover, there are many missing values for many features of the data. Therefore, we need machine learning models that can handle both data characteristics. In this paper, we apply and analyze the accuracy of new ensemble learning called XGBoost for the problem of claim prediction. We also compare the performance of XGBoost with that of another ensemble learning, i.e., AdaBoost, Stochastic GB, Random Forest, and online learning-based method, i.e., Neural Network. Our simulations show that XGBoost gives better accuracies in term of normalized Gini than other methods.

ACKNOWLEDGMENTS

This work was supported by Universitas Indonesia under PITTA 2018 grant. Any opinions, findings, and conclusions or recommendations are the authors' and do not necessarily reflect those of the sponsor.

References

[1] Hartwig, R.P., Lynch, J., Weisbart, S. (2016). More Accidents, Larger Claims Drive Costs Higher. Insurance Information Institute, New York.

[2] Weerasinghe, K.P.M.L.P., Wijegunasekara, M.C. (2016). A Comparative Study of Data Mining Algorithms in the Prediction of Auto Insurance Claims. In European International Journal of Science and Technology, vol. 5 no. 1, pp. 47-54

[3] Smith, K. A., Willis, R. J. (2017). An analysis of customer retention and insurance claim patterns using data mining: a case study. In Journal of the Operational Research Society, vol. 51, pp. 532-541

[4] L'heureux, A., Grolinger, K., Carrets, M. A. M. (2017). Machine Learning with Big Data: Challenges and Approaches. IEEE Access, vol. 5, pp. 7776-7796.

[5] Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Knowledge Discovery and Data Mining, 22nd ACM SIGKDD Conference on* (pp. 785-794)

[6] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer

[7] Bai, B. M., Mangathayaru, N., Rani, B.P. (2015). An approach to Find Missing Values in Medical Datasets. In *Engineering and MIS, 2015, ICEMIS 2015, The International Conference on* (article no. 70), ACM.

[8] Gajawada, S., Toshniwal, D. (2012). Missing Value Imputation Method Based on Clustering and Nearest Neighbours. In *International Journal of Future Computer and Communication*, vol. 1, no. 2, pp. 206-208.

[9] Zhang, S., Zhang, J., Zhu, X., Qin, Y., Zhang, C. (2008). Missing Value Imputation Based on Data Clustering. In *Gavrilova M.L., Tan C.J.K. (eds) Transactions on Computational Science I. Lecture Notes in Computer Science*. Springer. Berlin, Heidelberg.

[10] Fong, S., Luo, Z. (2013). Incremental Learning Algorithms for Fast Classification in Data Stream. In *Computational and Business Intelligence, 2013. 2013 International Symposium on* (pp. 186-190). IEEE.

[11] Dietterich, T. (2000). Ensemble Method in Machine Learning. Multiple Classifier System, vol. 1857. Springer, London.

[12] Genuer, R., Poggi, J.-M., Tuleau-Malot, C., Villa-Vialaneix, N. (2017). Random Forest for Big Data. *Big Data Research*, vol. 9, pp. 28-46.

[13] Aler, R., Galván, I.M., Ruiz-Arias, J.A., Gueymard, C.A. (2017). Improving the separation of direct and diffuse solar radiation components using machine learning by gradient boosting. In *Solar Energy* vol. 150, pp. 558-569.

[14] Volkovs, M., Yu, G. W., Poutanen, T. (2017). Content-based neighbor models for cold start in recommender systems. In *Recommender Systems Challenge 2017, RecSys Challenge '17, Proceedings*, article no. 7.

[15] Hand, D.J., Till, R.J. (2001). A Simple Generalization of the Area Under the ROC Curve for Multiple Class Classification Problems. In *Machine Learning*, vol. 45, no. 2, pp. 171-186.

[16] Little, R.J.A., Rubin, D.B. (2014). Missing-data Patterns. In *Statistical Analysis with Missing Data* (pp. 4-10). John Wiley & Sons, New Jersey.

[17] Tsai, C-F., Li, M-L., Lin, W-C. (2018). A class center-based approach for missing value imputation. In *Knowledge-Based Systems*, vol. 151, pp. 124-135.

[18] Zhang, Z. (2016). Missing data imputation: focusing on single imputation. In *Annals of Translational Medicine*, vol. 4, article 9.

[19] Deb, R., Wee-Chung, A. (2016). Missing value imputation for the analysis of incomplete traffic accident data. In *Information Sciences*, vol. 339, pp. 274-289.

[20] Ramezani, R., Maadi, M., Khatami, S.M. (2017). A novel hybrid intelligent system with missing value imputation for diabetes diagnosis. In *Alexandria Engineering Journal* (in press).

[21] Liu, Z-G., Pan, Q., Dezert, J., Martin, A. (2016). Adaptive imputation of missing values for incomplete pattern classification. In *Pattern Recognition*, vol. 52, pp. 85-95.

[22] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. In *Annals of statistics*, vol 29 no. 5, pp. 1189-1232.

[23] Freund, Y., Schapire, R.E. (1999). A short introduction on boosting. In *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771 – 780