International Journal of Control Theory and Applications

ISSN : 0974–5572

© International Science Press

Volume 9 • Number 40 • 2016

Experimenting XGBoost Algorithm for Prediction and Classification of Different Datasets

Ramraj S[a] Nishant Uzir[b] Sunil R[b] and Shatadeep Banerjee[b]

[a]*Asst. Prof. Department of Software Engineering, SRM University, KTR*

E-mail: ramraj.s@ktr.srmuniv.ac.in

[b]*UG Student, Department of Software Engineering, SRM University, KTR*

E-mail: nishant_bhagaban@srmuniv.edu.in, sunil_raman@srmuniv.edu.in, shatadeep_banerjee@srmuniv.edu.in

Abstract: Gradient boosting algorithm [1] was developed for very high predictive capability. Still its adoption was very limited because the algorithmrequires one decision tree to be created at a time in order to minimize the errors of all previous trees in the model. So it took a large amount of time to train even those models that were small in size. Then came a new algorithm called eXtreme Gradient Boosting (XGBoost)[2]which changed the way gradient boosting was done. In XGBoost, individual trees are created using multiple cores and data is organized in order to minimize the lookup times. This decreased the training time of models which in turn increased the performance. This research study strives to create a quantitative comparison of the accuracy and speed of XGBoost algorithm in multi-threaded single-system mode and Gradient Boosting with different datasets.

Keywords: *XGBoost, Gradient Boosting, multithreading, feature extraction.*

1. INTRODUCTION

The concept of boosting came to limelight when it was examined whether a “weak learner” could be made a “better learner” by using some kind of modifications. From statistics point of view, this process was similar to creating a “good hypothesis” from a relatively “poor hypothesis”. According to Jason Brownlee, author of [4], a poor learner or a “weak hypothesis” is a model whose performance is slightly better than random chance. Hypothesis boosting involves the idea of filtering the observations. Those observations which the weak learner can handle is left as it is and those observations that the weak learner cannot handle are focused on. According to [5], “The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified. … Note, however, it is not obvious at all how this can be done…”

The first application of boosting that was successful was Adaptive Boosting or AdaBoost[6] for short. According to [7], “Boosting refers to this general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb.” In the AdaBoost algorithm, the weak learners were given more weights and the strong learners were given less weights and this weight was changed repeatedly until a proper model was found which could correctly classify the given samples. When a prediction was needed to be done, the majority vote of the weakest learners' prediction was taken and the corresponding weight was chosen for the weight of the ultimate prediction.

This AdaBoost algorithm was then modified into a set of another statistical algorithms called ARCing algorithms. These algorithms used a process called "arcing". Arcing is an acronym for Adaptive Reweighting and Combining. Each step in an arcing algorithm consists of a weighted minimization followed by a re-computation of the classifiers and weighted input. This method was again modified to create a numerical optimization model called gradient boosting where the main function of the algorithm is to minimize the loss function by using a process that is somewhat similar to gradient descent. This model was described as a "stage-wise additive model". This is because one new "weak learner" is added at one time and the other "weak learners" are left unchanged.

The concept of gradient boosting involves basically three steps. First, a proper differentiable loss function should be identified that is suitable for the given problem. One benefit of the gradient boosting model is that for different loss functions, new algorithms are not required to be derived; it is enough that a suitable loss function be chosen and then incorporated with the gradient boosting framework. Second, a weak learner is created to make the predictions. In gradient boosting a decision tree is chosen as a weak learner. Specifically, regression trees are used that produces real value output for splits and whose output can be added together, allowing subsequent outputs of different models to be added. This approach enables the improvement of the residuals in the predictions leading to more precise predictions. The trees are created in a greedy manner and often certain constraints are imposed in order to ensure that the weak learners continue to be weak learners and still the trees can be created using a greedy approach. Third, creation of an additive model to add up the predictions of the weak learners so as to reduce the loss function. This process of adding the trees happens one at a time. The output produced in the new tree is then added to the output of the pre-existing sequence of trees in order to improve the final output of the model. This process stops once the proper optimized value for the loss function is reached.

XGBoost also has gradient boosting at its core. However, the difference between simple gradient boosting algorithm and XGBoost algorithm is that unlike in gradient boosting, the process of addition of the weak learners does not happen one after the other; it takes a multi-threaded approach whereby proper utilization of the CPU core of the machine are utilized, leading to greater speed and performance.

Apart from that, there is sparse aware implementation which also involves automatic handling of missing data values, then block structure to support the parallelization of tree construction, and the process of continued training so that one can further boost an already fitted model on new data.

It is to be noted that XGBoost has been seen to dominate structured or tabular datasets on classification and regression and predictive modeling problems.

2. DATASETS

2.1. Pima Indians Diabetes Dataset [8]

As per the source it has been collected from, all the patients are females, at least 21 years old, of Pima Indian heritage. Attribute Information:

2.1.1. Number of times pregnant

2.1.2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test

2.1.3. Diastolic blood pressure (mm Hg)

2.1.4. Triceps skin fold thickness (mm)

2.1.5. 2-Hour serum insulin (mu U/ml)

2.1.6. Body mass index (weight in kg/(height in m)^2)

2.1.7. Diabetes pedigree function

2.1.8. Age (years)

2.1.9. Class variable (0 or 1)

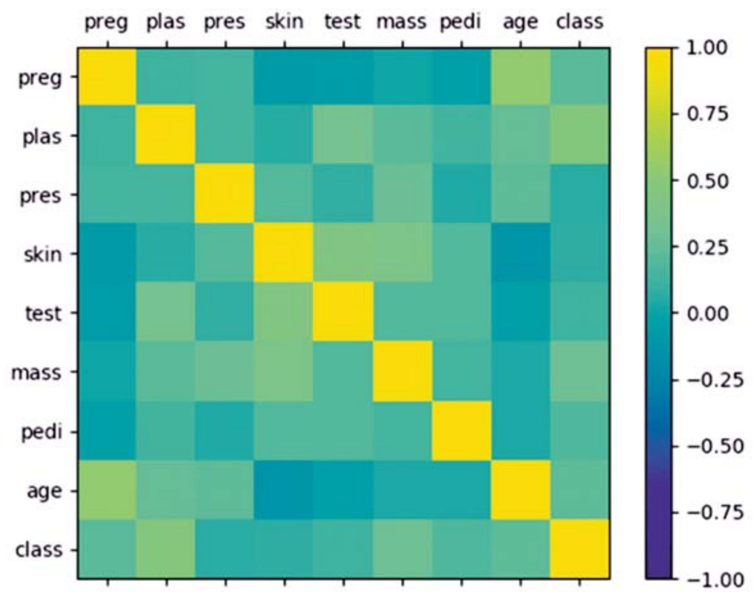

Figure 1: Correlation matrix of Pima Indians diabetes dataset

2.2. Airfoil Self-Noise Dataset [9]

According to the source website, the NASA data set comprises different size NACA 0012airfoils at various wind tunnel speeds and angles ofattack. The span of the airfoil and the observer position were the same in all of the experiments.Attribute information:

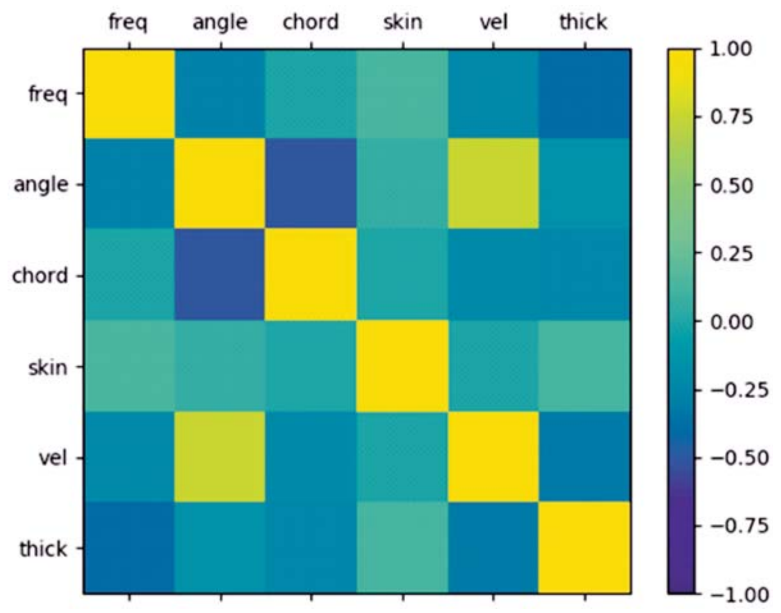

Figure 2: Correlation matrix of Airfoil Self

2.2.1. Frequency, in Hertz

2.2.2. Angle of attack, in degrees

2.2.3. Chord length, in meters

2.2.4. Free-stream velocity, (m/s)

2.2.5. Suction side displacement thickness, in meters.

2.2.6. Scaled sound pressure level, indecibels

2.3. Banknote Authentication Dataset [10]

Data were extracted from images that were taken from real and forged specimens. For digitization, a specialized industrial camera was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object, gray-scale images with a resolution of about 660 dpi were gained. According to the UCI Machine Learning Repository site, Wavelet Transform tool were used to extract features from the images. Attribute information:

2.3.1. Variance of Wavelet Transformed image (continuous)

2.3.2. Skewness of Wavelet Transformed image (continuous)

2.3.3. Kurtosis of Wavelet Transformed image (continuous)

2.3.4. Entropy of image (continuous)

2.3.5. Class (integer)

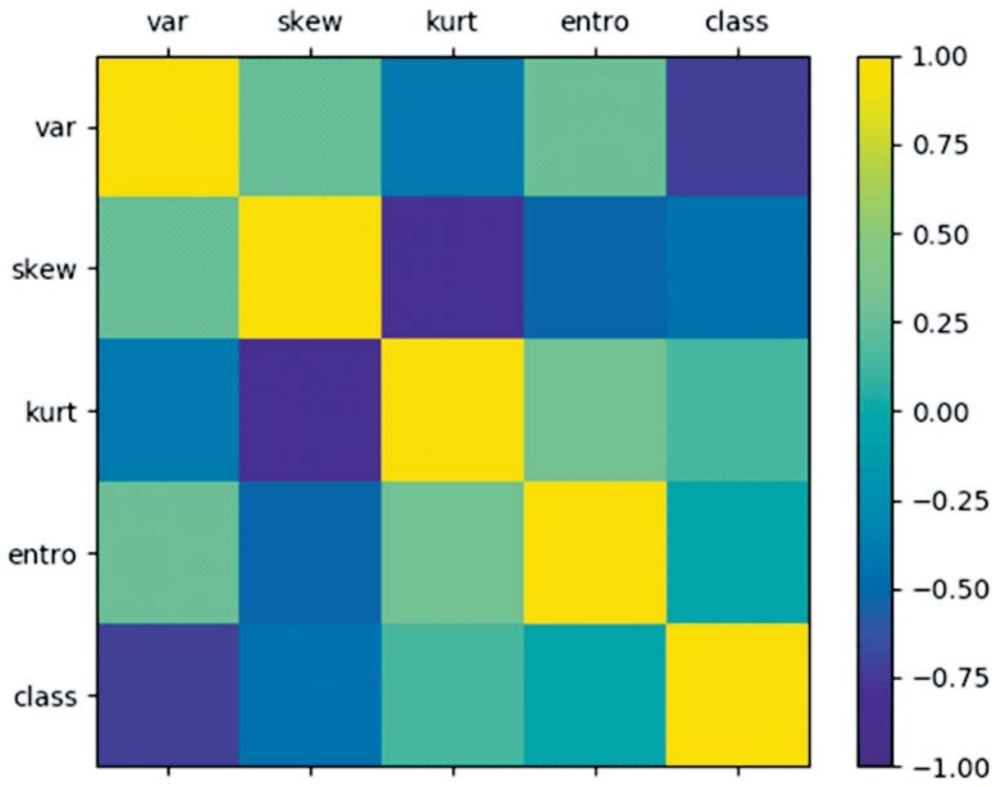

Figure 3: Correlation matrix of banknote authentication dataset

2.4. NIWE dataset [11]

This dataset was collected from the National Institute of Wind Energy, Government of India. The main objective here is to calculate the wind speed from the other given parameters like direction, surface temperature and surface pressure.Attribute information:

2.4.1. DateTime(year-month-day hour:minute)

2.4.2. Direction

2.4.3. Surface temperature

2.4.4. Surface pressure

2.4.5. Speed

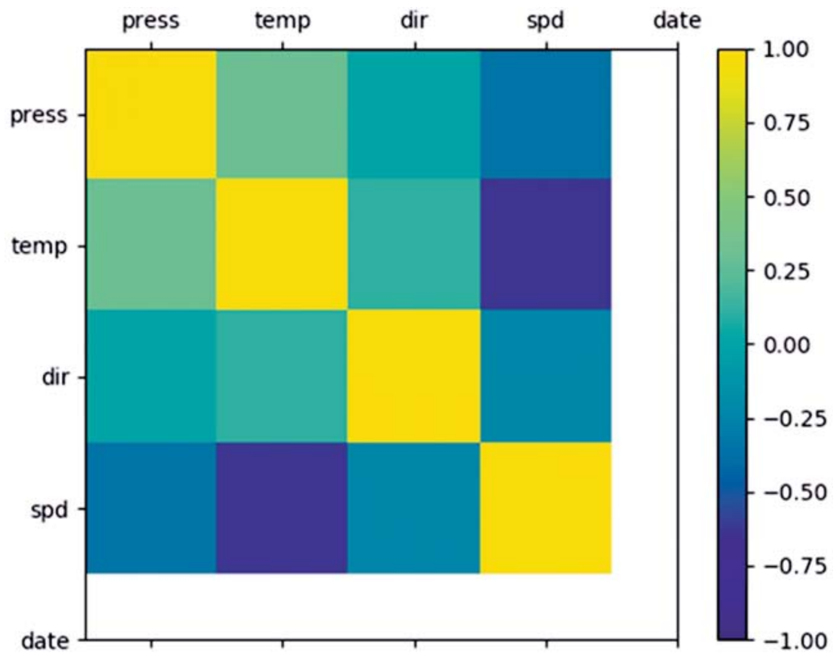

Figure 4: Correlation matrix of NIWE dataset

3. RELATED WORKS

In[12], *n* number of disjoint subsets were created and then a decision tree was trained on each of those disjoint sets. The individual learners are then combined and then a single decision tree is created. However, there are significant complexities in attempting such an approach. In their approach decision trees at each of n nodes were converted to rules and the rules were then combined into a single rule set. This single rule set was then used to classify unseen examples. They used the Pima Indians diabetes dataset along with the Irish dataset. Using 10-fold cross validation, they got an accuracy level of 73.33% and a standard deviation of 4.66%.

2.In [13], a feed forward multi-layer perceptron was used to create a prediction model for a turbulent boundary layer noise mechanism. A dataset similar to the airfoil self-noise dataset was used for this purpose. The network was trained and tested with 5, 10, 15 neurons. With 5 neurons, the training error was recorded as 6.722 and the validation error as 15.093. With 10 neurons, the training error was recorded as 3.731 and the validation error as 12.094. With 15 neurons, the training error was recorded as 2.500 and the validation error as 19.866. The optimal network architecture was determined as 5:10:1 and then again trained for 5000 epochs. It was discovered that at lower frequency values, the model followed the experimental data closely but at higher frequency values, the prediction is lower than the experimental values.

3. In [2], four datasets namely Allstate dataset, Higgs Boson dataset, Yahoo LTRC dataset, Criteo dataset were used. In the Allstate dataset the likelihood and cost of an insurance claim was to be predicted provided different risk factors were given. It was found out that the XGBoost algorithm was 10 times faster than the fastest algorithm among the other algorithms selected. In the Higgs Boson dataset, it is required to classify whether an event corresponds to the Higgs Boson. It was found out that when using XGBoost, the time taken by each tree was 0.6841 sec, which was the highest among all algorithms used. Also the area under curve(AUC) was calculated to be 0.8304, which was highest among all other algorithmsused. In the Yahoo LTRC dataset, the main objective is to rank the documents according to relevance of the query. It was found out that when using XGBoost, the time taken by each tree was 0.826 sec, which was the highest among all algorithms used. Also the normalized discounted cumulative gain(NDCG) was calculated to be 0.7892, which was quite decent. In the Criteo dataset, it was required to create a model regarding the scaling property of the system in the out-of-core and the distributed settings. It was found out that XGBoost executed 10 times more than spark per iteration and 2.2 times H2O's optimized version.

4. APPROACH

4.1. Pre-processing the datasets

In this research study, none of the datasets required pre-processing as each dataset contained only numeric data and none of the datasets had any missing values. As a result, no pre-processing was required.

4.2. Execution

The XGBoost algorithm has been executed in python in an i5 system having 4 cores. The code for the execution of the algorithm on all the four datasets has been made available in the GitHub repository[13]

4.3. Evaluation

Two methods were chosen for the purpose of evaluation of the models viz. train and test sets method and the *k*-fold cross-validation model.

The train and test sets model is one of the simplest models available where we split the entire dataset into training set and testing set. This method is particularly useful when the dataset involved is very large because we divide the training and testing sets beforehand. This method is important when the algorithm involved is slow in the training process. However, one disadvantage of this method is that the variance of the output can be high which can affect the overall accuracy of the model.The cross validation model is one where instead of one train-test set, *k* number of sets are created called "folds" and then *k*-1 folds are taken for training and the k^{th} fold is taken for testing. This is repeated until all the "folds" act as "test folds". The final result is the mean of all the individual results of all the "test folds". This provide a much better and accurate result as the model is trained multiple times on different data. The value of *k* is to be taken in such a way that the size of each "fold" is large enough so that there is proper training of the model in each repetition. Here, we have used the value of *k* to be 10.

5. DISCUSSION OF RESULTS

The following result was derived after executing XGBoost algorithm on the four datasets:

In terms of performance:

1. Pima-Indians-Diabetes dataset

   Type of problem: Classification

Table 1
Performance comparison for dataset[8]

Training set & testing set evaluation(GB)	*Training set & testing set evaluation(XGBoost)*	*10-fold cross validation model evaluation(XGBoost)*
72.05% (*acc*)	77.95% (*acc*)	76.69% (*acc*), 7.11% (*sd*)

2. Airfoil self-noise dataset

   Type of problem: Regression

Table 2
Performance comparison for dataset[9]

Training set & testing set evaluation(GB)	*Training set & testing set evaluation(XGBoost)*	*10-fold cross validation model evaluation(XGBoost)*
7.96 (*mse*)	7.91 (*mse*)	64.92% (*acc*), 23.98% (*sd*)

3. Banknote authentication dataset

   Type of problem: Classification

Table 3
Performance comparison for dataset[10]

Training set & testing set evaluation(GB)	*Training set & testing set evaluation(XGBoost)*	*10-fold cross validation model evaluation(XGBoost)*
99.78% (*acc*)	98.45% (*acc*)	99.56% (*acc*), 0.58%(*sd*)

4. NIWE dataset

   Type of problem: Regression

Table 4
Performance comparison for dataset[11]

Training set & testing set evaluation(GB)	*Training set & testing set evaluation(XGBoost)*	*10-fold cross validation model evaluation(XGBoost)*
2.55 (*mse*)	2.61 (*mse*)	65.21% (*acc*), 66.38% (*sd*)

In terms of speed :

Table 5
Speed comparison for all datasets

Dataset	*Execution time for training set & testing set evaluation(GB)*	*Execution time for training set & testing set evaluation (XGBoost)*
Pima Indians Diabetes Dataset	0.1012 sec	0.0117 sec
Airfoil Self-Noise Dataset	0.4900 sec	0.0843 sec
Banknote Authentication Dataset	0.1476 sec	0.0817 sec
NIWE dataset	0.1084 sec	0.0105 sec

The index for the acronyms have been provided below:

Table 6
Index table

Abbreviation	*Full form*	*Scikit-learn function*
acc	Accuracy score	accuracy_score()
mse	Mean squared error	mean_squared_ error()
sd	Standard deviation	std()
GB	Gradient Boosting	GradientBoostingClassifier() or GradientBoostingRegressor()

Thus we can clearly see that in case of the pima Indians diabetes dataset, for XGBoost, the accuracy is 77.95% in case of the training and testing set evaluation method. However, the accuracy comes out to be 76.69% in case of the 10-fold cross validation method with a standard deviation of 7.11%.Then, in case of GB, the accuracy is 72.05%. In case of the airfoil self-noise dataset, the mean squared error is 7.91 in case of the training and testing set evaluation method. However, the accuracy comes out to be 64.92% in case of the 10-fold cross validation method with a standard deviation of 23.98%. Then, in case of GB, the mean squared error is 7.96. In case of the banknote authentication dataset, for XGBoost, the accuracy is 98.45% in case of the training and testing set evaluation method. However, the accuracy comes out to be 99.56% in case of the 10-fold cross validation method with a standard deviation of 0.58%. For GB, the accuracy is 99.78%. In case of the NIWE dataset, for XGBoost, the mean squared error is 2.61 in case of the training and testing set evaluation method. The accuracy comes out to be 65.21% in case of the 10-fold cross validation method with a standard deviation of 66.38%. For GB, the mean squared error is 2.55.

In terms of the speed, in pima Indians diabetes dataset, for GB, the execution time is 0.1012 sec whereas for XGBoost, the execution time is 0.117 sec. In case of airfoil self-noise dataset, for GB, the execution time is 0.4900 sec, but for XGBoost it is 0.0843 sec. In case of banknote authentication dataset, for GB, the execution time is 0.1476 sec but for XGBoost it is 0.0817 sec. In case of NIWE dataset, for GB, the execution time is 0.1084 sec but for XGBoost it is 0.0105 sec.

It is to be noted that in case of regression, in training and test set evaluation, we are using mean squared error while in case of classification, we are using the accuracy score. Also, more the accuracy value, more is the accuracy of the model. But less the mean squared error value, more is the accuracy of the model.

It is also to be noted that since it is just a comparative study, this research study does not take into account the various tuning techniques available, as a result of which, the results obtained might not be the most accurate results. Thus, it might be possible that some other parties might claim to have got better results for the corresponding datasets using similar techniques and algorithms. But we can claim that the results obtained in this research study are valid and accurate enough to create a comparison between GB and XGBoost, both in terms of performance and speed.

5.1. Outcomes

The outcome of executing the gradient boosting tree is that we can get a knowledge as to which attribute contribute the most towards achieving the result. In short the feature importance can be found out and plotted. The feature importance graphs of the different datasets have been plotted below:

1. **Feature importance plot for Pima Indians diabetes dataset:** From the feature importance plot, we can clearly see that feature 5(body-mass index) is the most important feature which has contributed towards the prediction of the results. Followed by feature 1(plasma glucose concentration) and feature 6(diabetes pedigree function). The least important feature is feature 3(triceps skin fold thickness).

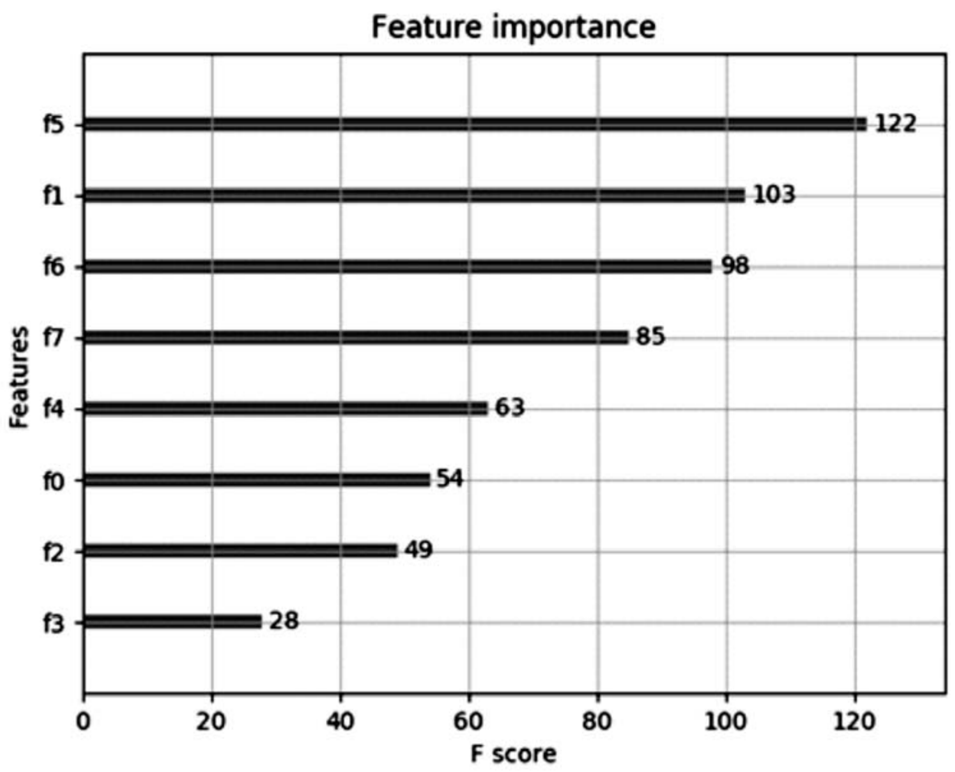

Figure 5: Feature importance plot of pima-indians-diabetes dataset

2. **Feature importance plot for airfoil self-noise dataset:** In the feature importance plot of the above stated dataset, we can see that feature 0(frequency) is the most important feature. Followed by feature 4(suction side displacement thickness). The least important feature is feature 3 (free-stream velocity).

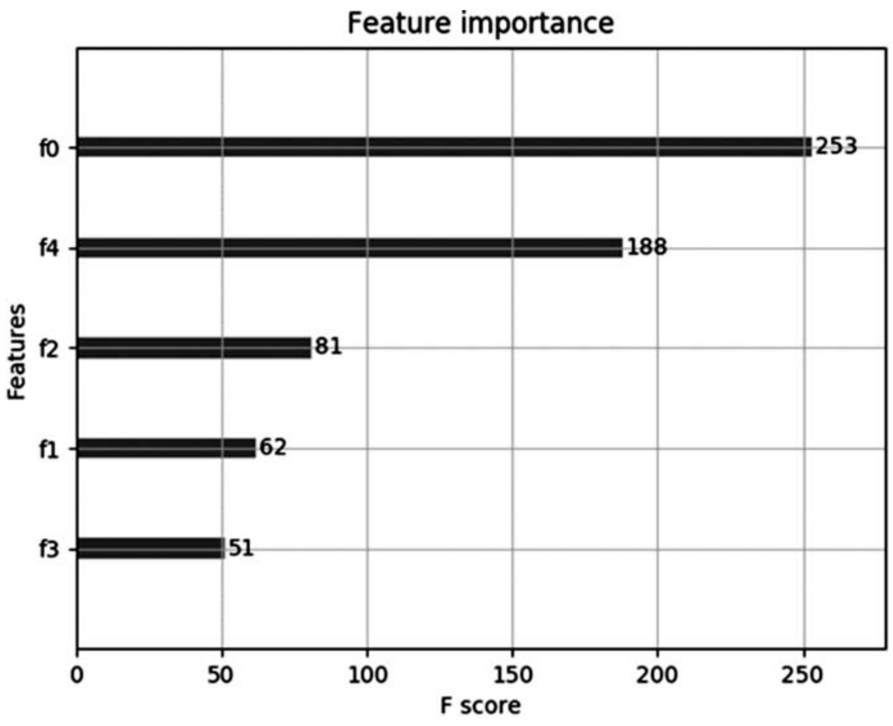

Figure 6: Feature importance plot for airfoil self-noise dataset

3. **Feature importance plot for banknote authentication dataset:** Here, feature 1(skewness) is the most important feature. Followed by feature 3(entropy of image). The least important feature is feature 0(variance).

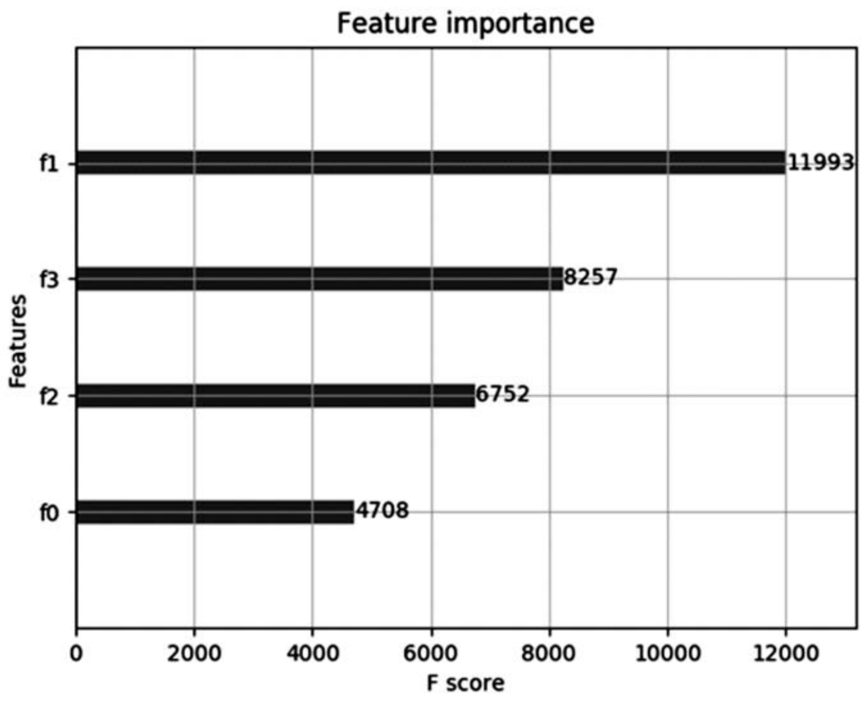

Figure 7: Feature importance plot for banknote authentication dataset

4. **Feature importance plot for NIWE dataset:** In the feature importance plot provided above, feature 0(direction) is the most important feature. The next important features are feature 1(surface temperature) and feature 2(surface pressure), which contribute somewhat equally to the prediction of wind speed. It is to be noted that date and time has no role in the determination of the wind speed.

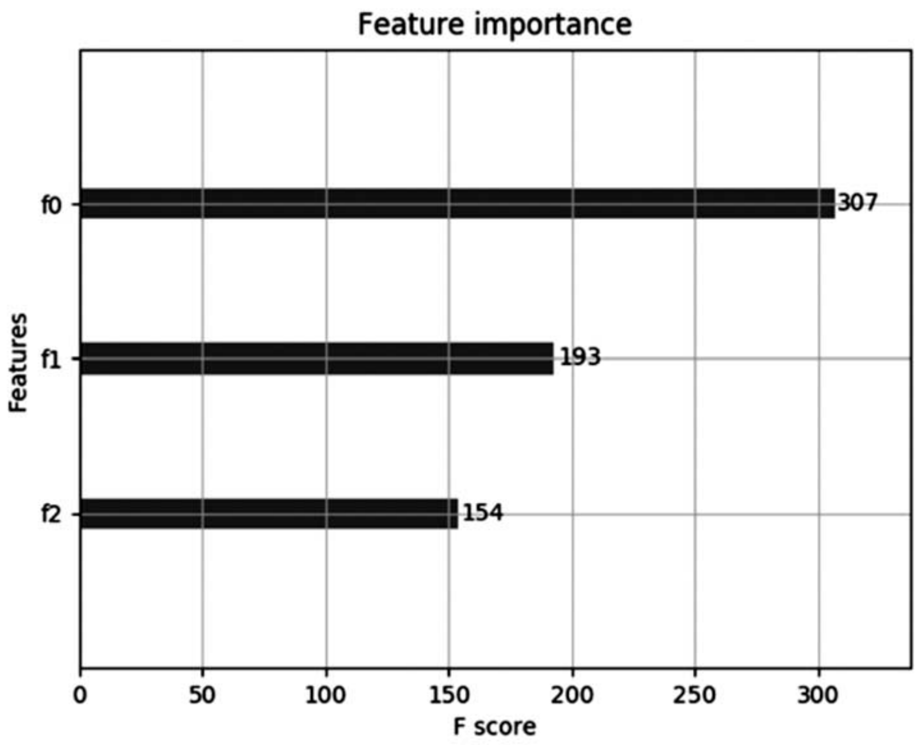

Figure 8: Feature importance plot for NIWE dataset

Thus, in this way, one can have an idea as to which features in a particular dataset contribute the most towards the arrival of a particular prediction.

6. CONCLUSION AND FUTURE WORK

Thus, in this research study we have given a detailed comparison of the accuracy and speed of execution of the XGBoost algorithm and the Gradient Boosting algorithm with different datasets performing different sets of operations viz. classification and regression based on the requirements of the datasets. It can be seen that the accuracy might not always be higher for the XGBoost algorithm when compared to the Gradient Boosting algorithm, but in case of speed of execution, XGboost has, almost always, been seen to be superior due to the application of multithreading.

However, the datasets that were chosen for the experiment did not have any missing data. So, the same results cannot be expected with a dataset containing missing values and noisy information.Therefore, in our future work we will examine the accuracy and speed of execution of the XGBoost algorithm with datasets that would have a variety of data types and missing data as well. Also, we would stress on the process of tuning the XGBoost algorithm for more appropriate results. In this process we will create robust models that would be able to train on real-world datasets where the occurrence of anomalies and noise are inevitable. In addition, since we are ushering into the age of BigData, we would try executing XGBoost algorithm using in-memory distributed systems like Apache Spark.

REFERENCES

[1] Greedy Function Approximation: A Gradient Boosting Machine, Jerome H. Friedman, IMS 1999 Reitz Lecture, February 24, 1999

[2] XGBoost : A scalable tree boosting system, Tianqi Chen, Carlos Guestrin, March 9, 2016, arXiv:1603.02754[cs.LG]

[3] XGBoost with Python, Jason Brownlee, Machine Learning Mastery

[4] Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World, Leslie Valiant, ISBN-13: 978-0465060726

[5] The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) 2nd Edition by Trevor Hastie, Robert Tibshirani, Jerome Friedman.ISBN-13:978-0387848570

[6] A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, Yoav Freund and Robert E. Schapire, December 19, 1996, AT6T Labs, 180 Park Avenue, Florham Park, New Jersey 07932

[7] National Institute of Diabetes and Digestive and Kidney Diseases , Pima Indians Diabetes Data Set, https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[8] Thomas F. Brooks, D. Stuart Pope and Michael A. Marcolini, NASA. Airfoil Self-Noise Data Set, https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

[9] Volker Lohweg (University of Applied Sciences, Ostwestfalen-Lippe, volker.lohweg **'@'** hs-owl.de),Banknote Authentication Data Set

[10] National Institute of Wind Energy, Ministry of New and Renewable Energy, Government of India

[11] Combining decision trees learned in parallel, Lawrence O Hall, Nitesh Chawla, Kevin W Bowyer, Department of Computer Science and Engineering, ENB 118 University of South Florida

[12] A Neural Networks Approach to Aerofoil Noise Prediction, K. Lau, R. López, E. Oñate, International Center for Numerical Methods In Engineering, Publication CIMNE Nº-335, 2009

[13] https://www.github.com/nishantuzir/xgboost