Analysis of Titanic Disaster using Machine Learning Algorithms

Abdullahi Adinoyi Ibrahim, *Member, IAENG and Rabiat Ohunene Abdulaziz, Member, IAENG*

***Abstract*—Analysis of Titanic shipwreck is essential in order to understand the historical data. The correlation between the independent and dependent features was observed in order to determine features that may have impact on passenger survival. In this paper, we explored the Titanic data and four machine learning algorithms namely XGBoost, CatBoost, Decision trees, Random forests were implemented to predict survival rate of passengers. Some factors such as *"Age", "Gender", "Children"* played a key role towards the survival of Passengers. A comparative analysis between these algorithms was conducted and the models were evaluated by some metrics. Based on the analysis and metrics evaluation, XGBoost outperformed other implemented algorithms in this work.**

Index Terms — XGBoost, CatBoost, Random forest, Decision trees, Titanic prediction, Python, Data mining.

I. Introduction

MACHINE achine learning algorithms have enabled data analysts, data scientists, data engineers and others to uncover insights from historical data. One of the most famous shipwrecks in history is the Titanic disaster, which happened on April 15, 1912. This was due to the collision with an iceberg which ripped off many parts of the giant ship. Facts emerged to support the cause of the shipwreck, and various speculations in regards to the Titanic's passengers survival rate [1]. The data for this disaster have been collected and available at [2]. One of the very powerful machine learning algorithms that achieves state-of-the-art results in a variety of hands-on tasks is the Gradient Boosting [3]. For several years, this algorithm continues to exist as the primary method for learning problems with noisy data, heterogeneous features and complex dependencies [4], [5]. Machine learning algorithms have been widely used in different areas. Some applications of machine learning methods can be found in [6], [7], [8], [9], [10], [11], [12].

Some authors have studied this disaster using various approaches. Eric [13] performed comparative analysis between three machine learning algorithms; Naive Bayes, Support Vector Machine (SVM) and decision trees with little gap of 2.64% between his best and worst models. Cicoria [14] performed cluster analysis and decision tree algorithms. In order to determine survival of passengers, the Sex feature was suggested as the major or most important factor [1]. Trevor [15] applied random forests and decision trees classifiers on some selected features of Titanic data and one of his model's best accuracy was given as 78%. Singh [1] implemented Logistic Regression, Naive Bayes, Decision Tree, Random Forest to predict the survival of passengers. The implemented Logistic regression achieved the highest accuracy of 94.26%.

Our approach shall differ from previous works. The objective of this paper is to perform data science processess on the Titanic data, by exploring the data, perform data cleaning and transformation, build predictive models such as Catboost algorithm, XGBoost, decision trees and random forests from the available features. Identify some factors that impact passengers survival rate, then, some metrics to evaluate the implemented algorithms shall be carried out.

The rest of the paper is structured as follows: machine learning algorithms to be used are given in II. Exploratory and predictive models, results are given in III. and IV gives the conclusion to the paper.

Manuscript received January 09, 2020; revised July 11, 2020.

A.A. Ibrahim is with Department of Mathematical Sciences, Baze University, Abuja, Nigeria. abdullahi.ibrahim@bazeuniversity.edu.ng.

R.O. Abdulaziz is with Deparment of Energy Engineering, Pan African Uiveristy of Water & Energy Resources, university of Tlemcen, Algeria. rabiat.abdulaziz@aims-cameroon.org.

II. Materials and Algorithms

A. Random Forest

Random forest algorithm is a well known tree based ensemble learning technique and the type of ensemble used is bagging [16]. Random forests are different from standard trees, for the latter, every node is split using the best split among all variables. In this model, each node is split using the best among a subset of predictors randomly chosen at that node [17]. This additional layer of randomness makes it robust against over-fitting [18]. Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors [19].

Random Forests Algorithm

The random forest algorithm for classification and regression is given as

(i) Draw m_{tree} bootsrap samples from the initial data.

(ii) Grow an *unpruned* tree (classification or regression) tree, for every bootstrap samples, with the modification given as follow: rather than choosing the best-split among all predictors at each note, sample randomly n_{try} of the predictors and select the best-split from among those features. Bagging can be seen as a special case of random forests which can be obtained when $n_{try} = k$, number of predictors.

(iii) Predict new data by aggregating the predictions of the m_{tree} trees.

An estimate of the error area can be obtained as described in [17].

B. Decision Trees

Decision trees is one of the supervised learning algorithms that can be applied to both classification and regression problems [20]. We shall briefly consider regression and classification tree problems. There are two steps (as explained in [20]) for building a regression tree, given as follow

(i) Divide the set of possible values $X_1, \ldots, X_n$ for into $I-$distinct and non-overlapping regions, $R_1, R_2, \ldots, R_i$.
(ii) For each observation that falls into R_i, we make the same prediction, which is simply the mean of the response feature for the training sets in R_i.

In order to construct regions $R_1, \ldots, R_i$, we elaborate on step (i) above. In theory, $R_1, R_2, \ldots, R_i$ could have any shape or dimension. However, we choose to split the predictor space into high-dimensional boxes, for simplicity and for easy interpretation of predictive models. The goal is to find boxes $R_1, \ldots, R_i$ that minimizes the Residual Sum of Squares (RSS) and the mathematical expression is given by

$$\sum_{i=1}^{I} \sum_{j \in R_i} (y_j - \hat{y}_{R_i})^2 \tag{1}$$

Where $\hat{y}_{R_i})$ is the mean response of the training sets within the ith box.

The classification tree on the other hand predict a qualitative response variable. In a classification tree, we predict that every observations belongs to 'most frequently occurring' class of training sets in the region to which it belongs Since we intend to allocate an observation in a given region to the 'most frequently occurring' class of training sets in that region, the classification error rate is the part of the training sets in that region that do not belong to the most frequent class, as given by

$$E = 1 - \max_{l}(\hat{p}_{ml}) \tag{2}$$

where $\hat{p}_{ml}$ denotes the proportion of training samples in the mth region that are from lth class. However, it turns out that classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable. The *Gini index* which is defined mathematically as

$$G = \sum_{l=1}^{L} \hat{p}_{ml}(1 - \hat{p}_{ml}) \tag{3}$$

A measure of total variance across the L classes. Further details can be found in [20].

C. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is one of the boosted tree algorithms [16], which follows the principle of gradient boosting [21]. When compared with other gradient boosting algorithms, XGBoost makes use of a more regularized model formalization in order to control overfitting of data, which gives it better performance [16]. In order to achieve this, we need to learn functions h_i, with each containing structure of tree and leaf scores [22]. As explained in [26], Given a data with m-samples and n-features, $\mathcal{D} = \{(X_j, y_j)\}(|D| = m, X_j \in \mathbb{R}^n, y_j \in \mathbb{R})$ a tree ensemble model make use of L additive functions to predict the output

$$\hat{y}_j = \phi(X_j) = \sum_{l=1}^{L} h_l(X_J), \qquad h_l \in \mathcal{H} \tag{4}$$

where $\mathcal{H} = \{h(X) = w_q(X)\}(q : \mathbb{R}^n \to U, w \in \mathbb{R}^U)$ is the space of regression trees. q denotes the structure of each tree that maps a sample to its corresponding leaf index. U denotes number of leaves in the tree. Each h_l corresponds to independent structure of tree q and leaf weights w.

To learn the set of functions used in the model, the regularized objective is minimized as follows

$$\mathcal{L}(\phi) = \sum_{j} l(\hat{y}_j, y_j) + \sum_{l} \Omega(h_l), \tag{5}$$

$$\text{and} \quad \Omega(h) = \gamma U + \frac{1}{2}\lambda ||w||^2 \tag{6}$$

where l is differentiable convex loss function that measures difference between the target y_j and prediction $\hat{y}_j$. Ω penalizes the complexity of the model to avoid over-fitting. The model is trained in an additive way. A score to measure the quality of a given tree structure q is derived as

$$\hat{\mathcal{L}}^{(u)}(q) = -\frac{1}{2} \sum_{j=1}^{U} \frac{(\sum_{i=I_j} f_i)^2}{\sum_{i=I_j} g_i + \lambda} + \gamma U \tag{7}$$

where $f_i = \partial_{\hat{y}^{(u-1)}} l(y_i, \hat{y}^{(u-1)})$ and $g_i = \partial^2_{\hat{y}^{(u-1)}} l(y_i, \hat{y}^{(u-1)})$ are the gradient and second order gradient statistics, respectively. Further explanation can be obtained in [26].

D. CatBoost

Another machine learning algorithm that is efficient in predicting categorical feature is the CatBoost classifier. Catboost is an implementation of gradient boosting, which make use of binary decision trees are base predictors [3]. Suppose we observe a data with samples $D = \{(X_j, y_j)\}_{j=1,\ldots,m}$, where $X_j = (x_j^1, x_j^2, \ldots, x_j^n)$ is a vector of n features and response feature $y_j \in \mathbb{R}$, which can be binary (i.e yes or no) or encoded as numerical feature (0 or 1). Samples (X_j, y_j) are independently and identically distributed according to some unknown distribution $p(\boldsymbol{.}, \boldsymbol{.})$. The goal of the learning task is to train a function $H : \mathbb{R}^n \to \mathbb{R}$ which minimize the expected loss

$$\mathcal{L}(H) := \mathbb{E}L(y, H(X)) \tag{8}$$

where $L(\boldsymbol{.}, \boldsymbol{.})$ is a smooth loss function and (X, y) is a testing data sampled from the training data D.

The procedure for gradient boosting [21] constructs iteratively a sequence of approximations $H^t : \mathbb{R}^m \to \mathbb{R}, t = 0, 1, \ldots$ in a greedy fashion. From the previous approximation H^{t-1}, H^t is obtained in an additive process,such that $H^t = H^{t-1} + \alpha g^t$, with a step size α and function $g^t : \mathbb{R}^n \to \mathbb{R}$, which is a base predictor, is selected from a sets of functions G in order to reduce or minimize the expexted loss defined as

$$g^t = argmin_{g \in G} \mathcal{L}(H^{t-1} + g) \tag{9}$$

$$g^t = \arg\min_{g \in G} \mathbb{E}L(y, H^{t-1}(X) + g(X)) \quad (10)$$

Often times, the minimization problem is approached by the Newton method using a second-order approximation of $\mathcal{L}(H^{t-1} + g^t)$ at H^{t-1} or by taking a (negative) gradient step. Either of these functions are gradient descent [23], [24]. Further expalanation on CatBoost algorithm can be obtained in [3].

III. Design and Result

The data consist of 891 training samples/observations with 12 features (in table I) and 471 test examples as provided by kaggle [2].

A. Exploratory analysis

Exploratory analysis of the data was performed in order to get insights of the data. Furthermore, the summary statistics such as mean, standard deviation, minimum value, first and third quartiles, median and maximum values were also observed. Features of the data are described in table I. Plots generated from the exploratory are shown in figures $1 - 8$.

TABLE I
Feature description

Features	Description
PassengerID	An unique index for passenger rows.
Survived	Shows if the passenger survived or not.
Pclass	Ticket class. $1st$, $2nd$, and $3rd$
Name	Passenger's name
Sex	Passenger's sex. It's either Male or Female.
SibSP	No. of siblings/spouse aboard.
Parch	Number of parents/children aboard.
Ticket	Passenger Ticket Number
Fare	How much money the passenger has paid
Cabin	Passenger Cabin
Embarked	Port of boarding.
Age	Passenger's age.

Starting with figure 1, it was observed that children between the $0 - 5$ years had a higher chance of survival than adults.

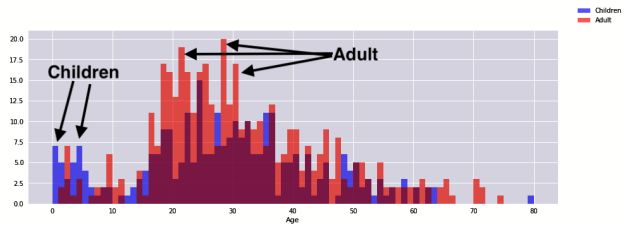

Fig. 1. Survival rate by Age

In figure 2, females in $1st$ class stood a hgiher chance of survival than male in all the classes. Children in $2nd$ class stand a better chance of survival rate than both male and female in $1st$ class.

Passengers embark on the Titanic at three different stops; QueensTown (Q), Cherbourg (C) and Southampton (S). From the data, Southampton may be the largest city among the point of embark (3). Most of the passengers were working class with few high borns. In Queenstown, almost everyone that embarked the Titanic were in $3rd$ class. This might be

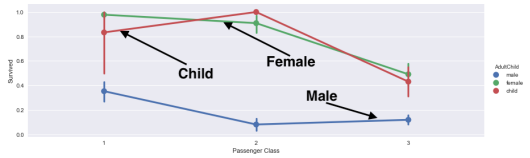

Fig. 2. Survival rate by passenger class

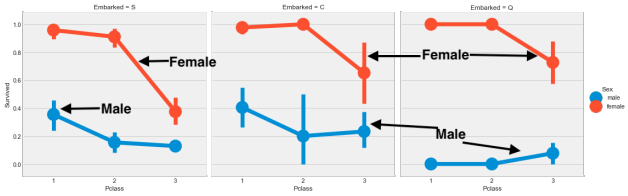

Fig. 3. Survival rate by point of embarked

that Queentown was a small city with little socio economic condition. Passengers that embark at Cherbourg dominated the $1st$ and $3rd$, this might be due to Cherbough consist of high born and working people.

In figure 3, female passengers in $1st$ class that boarded the ship at points S,C,Q had a better higher chance of survival than other classes. It was also observed that there is a reduction in survival rate from $2nd$ to $3rd$ class for all the points of boarding.

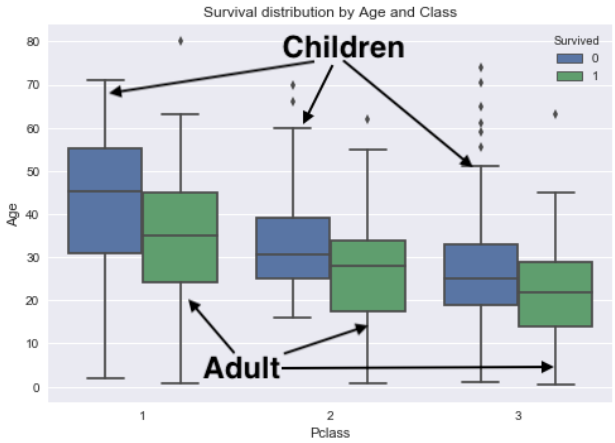

Fig. 4. Survival rate by Age & Passenger class

In figure 4, the younger population had a higher survival rate in all the classes.

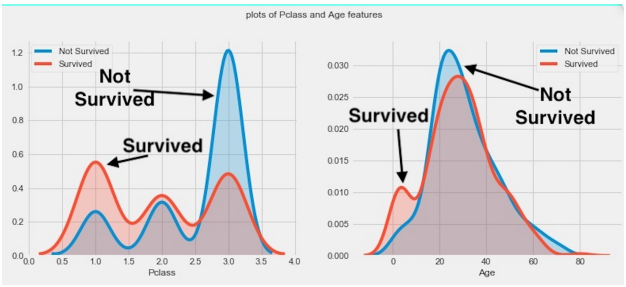

Fig. 5. Survival rate for the *Pclass* and *Age* features

From figure 5, the first plots shows Passengers in $1st$ Pclass had a higher survival rate and the plot on the right is the Age feature, combining both male and female, we can

see that children with age between 0 to 5 had a better chance of survival.

Figure 6 chances of survival decreases with increase in age of individuals. Also, People in $1st$ Class were given higher priority for rescue than other classes.

The violin plots in figure 7 shows that the number of children increases from $1st$ to $3rd$ classes. Also, chilren between age ≤ 10 had a higher chance of survival in all classes.

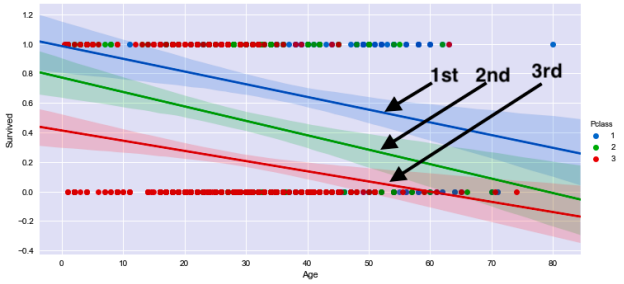

Fig. 6. Correlation between Age and Survival

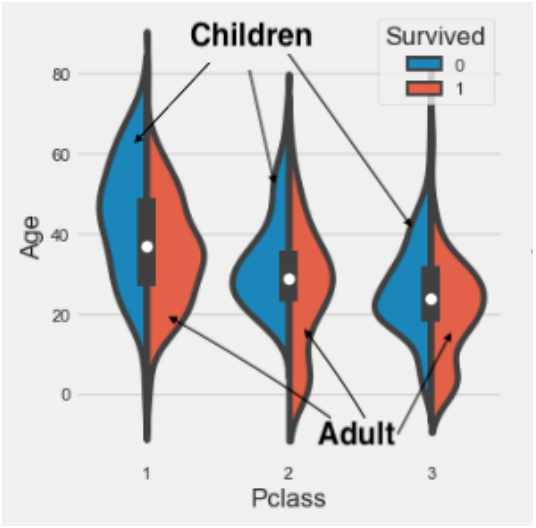

Fig. 7. Pclass against Age

In figure 8, the age range of an adult (both male & female) are uniformly equal as shown in the plot. For the children, there are two significant bumps; the main bump is at around $2-3$ year range and a slight bump is seen close to the age of 15.

B. Data pre-processing

After the initial exploratory, it was observed that the following features; *Age*, *Cabin*, *Embarked* had $177, 687, 2$ missing values or Not available value (NA's), respectively. First we need to perform data cleaning on the features with missing data.

All the NA's in our data are found in the numerical features, we started by replacing the NA's. Starting with "*Age*" features on a case-by-case basis. On the "*Age*" feature, we replaced the missing values randomly with mean $\pm$ standard deviation of the feature. For the "*Cabin*" features, we replaced the missing values with median. Suppose we had NA's on categorical features, we would have imputed

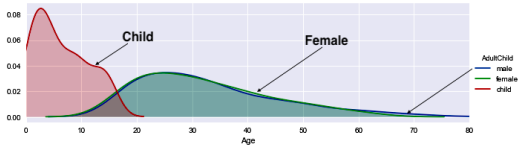

Fig. 8. Survival rate by Age

TABLE II
CONFUSION MATRIX

	Predicted		
		Negative	Positive
Actual	Negative	True Negative (TN)	False Negative (FN)
	Positive	False Positive (FP)	True Positive (TP)

using the mode of the feature. Some numerical features were replaced with median as it handles the presence of outliers unlike the mean imputation. Also, it is worthy to note that the data is balanced in ratio $49:51$.

C. Model validation technique

The training data provided by kaggle.com [2] was used in the training Model and the test sample provided was used to test the model. A grid-search was performed over tuning parameters, including regularization or penalty hyper-parameters, for each algorithm. The models were trained using their optimal configuration on the training dataset. The trained model from each algorithm was then used to predict and tested on the test samples. In the correlation between all features with the "Survived" feature, it was observed that 'Pclass' had the highest correlation of -0.338481 with "Survived".

D. Confusion Metrix

A confusion matrix contains information about actual and predicted classifications one by a classifier algorithm, which are the classifiers studied. Performance of such classifiers are commonly evaluated using the data in the matrix. The table II shows the confusion matrix for a classifier [25].

True Positive: The classifier predicted Survived and the passenger actually Survived.

True Negative: The classifier predicted Not Survived and the passenger actually did Not Survived.

False Positive: The classifier predicted Survived but the passenger actually did Not Survived.

False Negative: The classifier predicted Not Survived but the passenger actually Survived.

The confusion matrix can be interpreted as: the TN and TP are the correctly classified classes while FN and FP are the mis-classified.

E. Model evaluation metrics

The Model training time, model accuracy and memory utilized are some good metrics for comparing the performance of the classifiers. In addition, the Area under the Receiver Operating Characteristics Curve (ROC-AUC) is a performance metric for classification accuracy. The AUC is another metric which checks the performance of multiple-class classification accuracy [22].

Model accuracy is the proportion of the proportion of the correct predictions (True positive and True negative) from the total predictions defined given in (11)

$$Accuracy = \frac{TN + TP}{TP + TN + FP + FN} \times 100\% \qquad (11)$$

The True Positive Rate (TPR), also called the sensitivity or recall, is the proportion of correct positive predicted class from total positive class. The best sensitivity is 1.0 and worst is 0.0. The True Negative Rate (TNR), also called the specificity, is the proportion of correct negative predictions from the total number of negative classes. The best specificity is 1.0 and worst is 0.0. The TPR and TNR are given in (12).

$$TPR = \frac{TP}{FN + TP} \times 100, TNR = \frac{TN}{FP + TN} \times 100 \qquad (12)$$

Precision is the number of correctly predicted positive values out of the total number of positive classes, as given in (12). False positive rate (FPR) is the number of incorrect positive predictions out of the total number of negatives as in (13).

$$FPR = \frac{FP}{FP + TN} \times 100 \qquad (13)$$

Error rate is the proportion of of all incorrect predictions divided by total number of samples, given as

$$Errorrate = \frac{FP + FN}{FP + FN + TP + TN} \times 100 \qquad (14)$$

F. System specification

All classifiers were run on the jupyter notebook in python 3.7.4 on the linux 19.10 version. The codes were run on 8GB HP elite book, core $i5$.

G. Results and discussion

The result of the experiments is given in the table III with five (5) columns: lists of algorithms implemented (Algorithms), the model accuracy (scores), AUC for training model (AUC (training)), Average time to fit the model in python (Avg time to fit (s)), and average time for the model to score (Avg time to score (s))

TABLE III
COMPARISON OF ALGORITHMS

Algorithms	Scores	AUC (training)	Avg. time (fit)	Avg. time (score)
Decision Trees	88.12	0.80	0.001	0.0
Random Forest	88.12	0.85	0.152	0.01
Catboost	87.21	0.85	0.743	0.026
XGBoost	91.00	0.89	0.24	0.008

In table III, the four algorithms performance were evaluated using some metrics and compared. XGBoost achieved the highest accuracy of 91.00%. Model accuracy is the measure of how well a model predicts as defined in (11). False positive rate is a method of committing type I error in null hypothesis testing when conducting multiple comparisons. For the problem used in this paper, false positive rate is an important metric as it would be a disaster if the system predicts a passenger survived but in reality he does not survive. Catboost algorithms achieved the highest average time (python time) to fit the model.

TABLE IV
XGBOOST CLASSIFICATION

		Predicted	
		0	1
Actual	0	140 (TN)	21 (FN)
	1	20 (FP)	290 (TP)

Table IV shows the correctly classified and mis-classified classes for the XGBoost algorithm (Since this algorithm attained the highest score). The correctly classified classes summed up to 430.

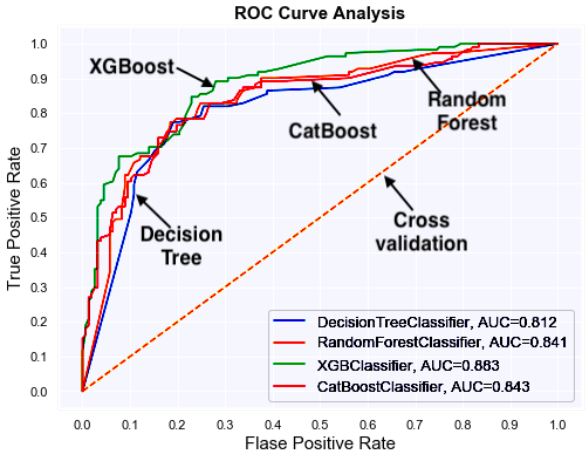

Fig. 9. AUC curves for the algorithms

In figure 10, one can observe that XGBoost and CatBoost classifiers are well calibrated than other implemented algorithms here. Figure 11 shows the proportion train vs test error rate. After training the models on the training set and predicted the probabilities on the test set. We then obtain the True positive rate (sensitivity), False positive rate $(1 - specificity)$ and AUC scores. From figure 9, XGBoost achieved the highest AUC value of 0.89 which is closer to 1 than other classifiers. Also, figure 12 shows the comparison between the four algorithms discussed. The names of the algorithms are written in short form as CART for decision trees, RF for random forest, XGB for XGBoost and CAT for CatBoost. Figure 12 shows the spread of the accuracy scores across each cross validation fold for each algorithm. Figure 12 is generated based on the mean accuracy and standard deviation accuracy of the algorithms.

Figure 12 would suggest that XGBoost is perhaps worthy of further study on this problem due to its performance. The result has been presented in table III which contains the model accuracies, AUC, average time to fit and score. The ROC curve is presented in figure 9 and model performance is shown in figure 12.

H. Impact of some factors on Passenger survival

From the exploratory analysis shown previously, some factors such as *"Age", "Gender", "Children"* plays a key role towards the survival of passengers.

(i) From the *Age* factor, we saw that priority was given to children and older people in rescue. In the Age and

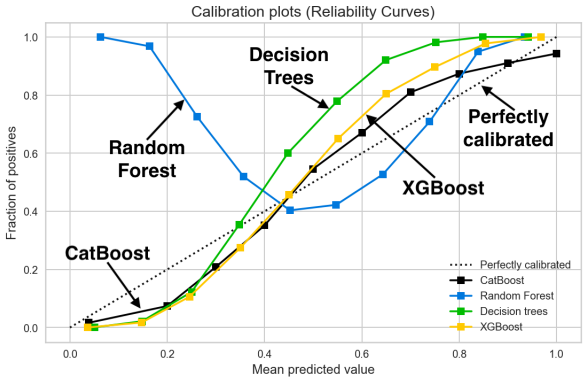

Fig. 10. Calibration plots

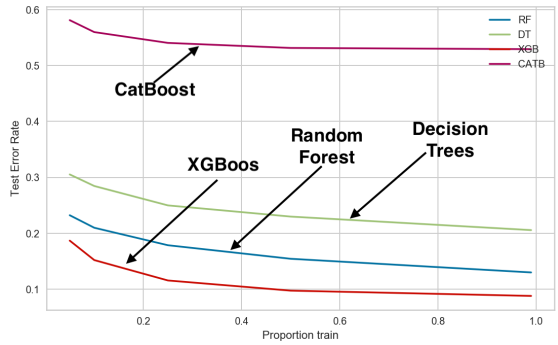

Fig. 11. Propoertion vs error rate

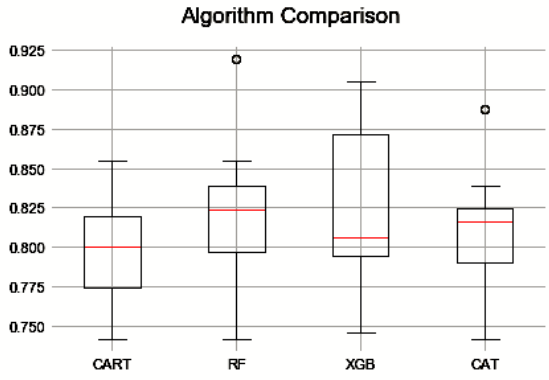

Fig. 12. Boxplots showing the distributions of the algorithms

Plass plot (Fig. 4), it was discovered that a passenger's class also played a significant role in his/her survival. Passengers in $1st$ class had a better chance of survival. Also, (Fig. 1) shows that the younger and the older people actually had a higher survival rate than the middle-aged passengers which acertain to the protocol maintained.

(ii) Another factor that plays a significant role towards passenger survival is their *Gender*. It was seen (Fig. 2) that females and children were first considered towards lifeboats before the male were considered. On the overall probability of passenger survival rate, the female population had a higher chance of survival (74%) than male.

(iii) The *children* factor also played an important role. Children were given priority in all the passengers class

IV. Conclusion

Based on the performance of the four mentioned algorithms; XGBoost, CatBoost, Decision trees, Random Forests, in this paper, XGBoost proved to be the best algorithm by outperforming other implemented algorithms for the Titanic classification problem since it achieved the highest accuracy. This implies that the number of correctly classified classes in XGBoost is higher than that of other implemented algorithms. Also, the AUC and boxplots values for XGBoost appear to be the highest as compared with other implemented algorithms. Based on our data, XGBoost appears to be a very good classifier.

Future work might consider cross validation. Cross validation could also be used to compute the model's accuracy based on different combinations of training and test samples. In addition, some other classifiers can also be applied.

References

[1] Analyzing Titanic disaster using machine learning algorithms-Computing, Communication and Automation (ICCCA), 2017 International Conference on 21 December 2017, IEEE.

[2] Kaggle.com, 'Titanic:Machine Learning form Disaster',[Online]. Available: http://www.kaggle.com/. [Accessed: 14- October- 2019].

[3] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In Advances in Neural Information Processing Systems (pp. 6638-6648).

[4] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning, pages 161–168. ACM, 2006.

[5] Y. Zhang and A. Haghani. "A gradient boosting method to improve travel time prediction". *Transportation Research Part C: Emerging Technologies*, 58:308–324, 2015.

[6] Jia, J. W., & Mareboyana, M.,"Machine learning algorithms and predictive models for undergraduate student retention," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2013 Vol I, WCECS 2013, 23-25 October, 2013, San Francisco, USA.

[7] Bogle, S., & Potter, W. "A machine learning predictive model for the jamaica frontier market". *In Proceedings of the 2015 Int'l Conference of Data Mining and Knowledge Engineering*. 2015.

[8] Susnjak, T., Barczak, A., & Reyes, N. "A decomposition machine-learning strategy for automated fruit grading". Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering and Computer Science 2013, 23-25 October, 2013, San Francisco, USA, pp819-825.

[9] Arafat MY, Hoque S, Xu S, Farid DM, "Machine learning for mining imbalanced data". *IAENG International Journal of Computer Science*, vol 46, no.2, pp332-348. 2019.

[10] Langovoy, M. "Machine learning and statistical analysis for brdf data from computer graphics and multidimensional reflectometry". *IAENG International Journal of Computer Science*,vol 42(1), 23-30. 2015.

[11] Lalis, J. T. "A new multiclass classification method for objects with geometric attributes using simple linear regression". *IAENG International Journal of Computer Science*, Vol 43(2), 198-203. 2016.

[12] Lei, C. U., Man, K. L., & Ting, T. O. "Using learning analytics to analyze writing skills of students: A case study in a technological common core curriculum course". *IAENG International Journal of Computer Science*, vol 41(3), 193-197. 2014.

[13] Eric Lam, Chongxuan Tang, "CS229 Titanic – Machine Learning From Disaster", *stanford university* 1-5, 2012.

[14] Cicoria, S., Sherlock, J., Muniswamaiah, M. and Clarke, L, "Classification of Titanic Passenger Data and Chances of Surviving the Disaster", *Proceedings of Student-Faculty Research Day*, CSIS, pp. 1-6, May 2014.

[15] Trevor Stephens. "Titanic: Getting Started With R - Part 3: Decision Trees". Available: http://trevorstephens.com/kaggle- Titanic-tutorial/r-part-3-decision-trees/. [Accessed: 15- December- 2019]. 2014

[16] R. Punnoose and P. Ajit, "Prediction of Employee Turnover in Organizations using Machine Learning Algorithms", *International Journal of Advanced Research in Artificial Intelligence (IJARAI)* , Vol. 5, No. 9, 1-5, 2016.

[17] A. Liaw and M. Wiener, "Classification and regression by randomForest", *R news*, 2(3), 18-22, 2002.

[18] L. Breiman, "Random forests". *Machine Learning* , vol 45(1), 5–32, 2001.
[19] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, An Introduction to Statistical Learning with Applications in R. *Springer textbook* 2013.
[20] G. James et al., An Introduction to Statistical Learning: with Applications in R, Springer Texts in Statistics 103, DOI 10.1007/978-1-4614-7138-7 8.
[21] J. H. Friedman, "Greedy function approximation: a gradient boosting machine", Annals of statistics, 1189-1232, 2001.
[22] S. Lessmann and S. Voß, "A reference model for customer-centric data mining with support vector machines", European Journal of Operational Research 199, 520–530, 2009.
[23] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. The annals of statistics, 28(2):337–407, 2000.
[24] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In Advances in neural information processing systems, pages 512–518, 2000.
[25] Kohavi, R. and Provost, F. Glossary of terms. Machine Learning—Special Issue on Applications of Machine Learning and the Knowledge Discovery Process. Machine Learning, 30, 271-274, 1998.
[26] T. Chen and C. Guestrin, "XGBoost: Reliable Large-scale Tree Boosting System", 2015.