A Comparison of the Performance of Machine Learning Models in Predicting Heart Disease

A. Barrientos Pena*, M. Evans†, P. Kulkarni‡ and S. Nikolaidou§
*35672724 †36059913 ‡36004026 §36052769

***Abstract*—This paper presents and evaluates the performance of the Logistic Regression and Gradient Boosting classifiers using a heart disease data set. The models were tested under similar conditions (data pre-processing and proportion of training and test observations were kept invariant) to determine which algorithm had greater accuracy in predicting whether a person is likely to have heart disease based on factors such as age, sex, chest pain, blood pressure, serum cholesterol, blood sugar and ECG. With an accuracy of 87%, the gradient boosting algorithm using XGBoost proposes the best accuracy compared to the Logistic Regression model. In conclusion, the two classifiers performed satisfactorily in predicting a subject having heart disease. The presented algorithms could be employed in future cardiology research or in diagnosis.**

***Index Terms*—Classification, Logistic Regression, Gradient Boosting, Heart Disease**

I. Introduction

Coronary heart disease was found to be the leading cause of death in the world in 2019 [11] and was responsible for 16 million deaths globally. Fatal cases of coronary heart disease have risen since 2000 likely due to population growth and ageing [12]. Heart disease is diagnosed by performing a sequence of tests which, depending on the patient, ranges from blood tests and exercise stress tests to CT scans. Given several factors such as age and pre-existing conditions, classification machine learning algorithms can help predicting a possible heart disease.

Classification refers to the use of machine learning algorithms that learn to assign a class in future observations based on training (past) data. Within this report, a data set consisting of 918 observations, a combination of data from 5 locations (originating from the UCI machine learning repository), was used to train and test binary classifiers to determine a patient's heart disease status. The data included variables of age, sex, resting blood pressure, cholesterol levels, fasting blood sugar, resting ECG, maximum heart rate, exercise angina and ST slope results, with a heart disease response variable. The considered classifiers were logistic regression and gradient boosting. Using a binary classifier as a diagnostic tool can give practitioners feedback and support the diagnostic process.

Binary classifiers in recent years have been used extensively for medical tests and are a relatively common application of machine learning methods. They can be used to give practitioners feedback and support diagnoses. This report will detail the pre-processing of the data, the methodology behind each classifier considered, followed by an assessment of accuracy to determine which classifier is most suited to predicting a patient's heart disease status from the variables considered.

II. Methodology

A. Pre-Processing

Commonly, raw data has inconsistencies. It can present outliers, duplicated observations, missing values, and so on. If the raw data is not cleaned, the machine learning results will have deficiencies and a lack of quality. Thus, data pre-processing is imperative to improve any subsequent analysis.

The pre-processing of the data commenced with the searching of missing values. The set did not present any. Since the features are heterogeneous (quantitative and qualitative), and the chosen machine learning algorithms depend on numeric data type, it was necessary to transform (encode) the categorical inputs into numeric. To meet this purpose, ordinal and dummy variable encoding techniques were used. The prior was used for the ordinal type, in which each unique category label is assigned to an integer value. These integers have a natural order, so the ranking of the categories remains the same. The dummy variable encoding method was implemented for the remaining qualitative features. As there isn't a known relationship between the categories, the ordinal encoding approach was unsuitable. By transforming the nominal features using the dummy variables technique, the performance of the classifiers was improved.

The cholesterol attribute presented inconsistencies; zero values were found. This is not possible, since cholesterol is present in all humans. Zeros were replaced by the median. The mean was not employed given their sensitivity to outliers. To create consistency amongst quantitative features and achieve better performance results in the classifiers, observations were standardised. Standardisation of the data was also used for the detection of outliers. Inputs outside the [-3, 3] range were dropped. For doing so, a normal distribution was assumed. The original data set, containing 918 observations, was reduced to 889. Reducing the features in the data set was not considered adequate, given that the number of covariates is relatively small.

Splitting the data is necessary for training and testing the different classification algorithms. To reduce overfitting (the model learning the noise and detail of the training data) or underfitting (observations not being sufficient to train the model) the classifiers, and to reduce the bias, 80% of the observations were utilized for training and 20% for testing. It is necessary to note that the proportion of each class in the response variable was not equal. In both sets, class 1 was predominant than class 0.

B. Logistic Regression

Logistic regression models the probability of the response variable to belong to a K class via linear functions in x. The model ensures that the probability lies within the [0, 1] range through the usage of the logistic function. The model is specified in terms of *K-1* log-odds or logit transformations [5], where for $k = 1, \ldots, K-1$:

$$Pr\left(G = k \mid X = x\right) = \frac{e^{\left(\beta_{k0}+\beta_k^T x\right)}}{1+\sum_{l=1}^{K-1} e^{\left(\beta_{l0}+\beta_l^T x\right)}},$$

$$Pr\left(G = K \mid X = x\right) = \frac{1}{1+\sum_{l=1}^{K-1} e^{\left(\beta_{l0}+\beta_l^T x\right)}}$$

The logistic regression is a linear classifier, where $\beta_{k0}+\beta_k^T x$ is a linear function known as logit. The estimated parameters are the betas (β). These are referred as the coefficients or predicted weights. This classifier is commonly used in biostatistical application where binary responses occur [1]. When *K = 2*, this model is simple, since there is only a single linear function,

$$Pr\left(X\right) = \frac{e^{(\beta_0+\beta_1 X_1+\ldots+\beta_r X_r)}}{1+e^{(\beta_0+\beta_1 X_1+\ldots+\beta_r X_r)}},$$

Where *r* is the number of inputs or covariates. The logistic regression model captures better the range of probabilities than a linear regression model and will always generate a *S-shaped* curve (sigmoid function) [7]. With further manipulation, the above expression is transformed to:

$$\frac{Pr(X)}{1\ -\ Pr(X)} = e^{(\beta_0+\beta_1 X_1+\ldots+\beta_r X_r)}$$

The $Pr(X)/(1-Pr(X))$ is known as the odds and can take on any value within 0 and ∞. Odds values near 0 indicate very low probabilities of success. On the contrary, values close to infinite demonstrate very high probabilities of success. In a linear regression model the predicted weights give the average change in the dependent variable (Y) associated with a one-unit increase in the covariates (X). The interpretation of the coefficients in the logistic regression model is different. Increasing X by one unit changes the log odds by β. Despite the value of X, if the estimators are positive, then increasing X will be associated with an increase in Pr(X). If the estimators are negatives, decreasing X will imply a decrease in Pr(X) [7]. The beta parameters need to be estimated using training data. According to James *et al* (2013), although (non-linear) least squares could be employed for finding the coefficients, the more general method of maximum likelihood is preferred; the statistical properties are better. The maximum likelihood estimation, identifies, through iterative cycles, the strongest linear combination of independent variables that increases the likelihood of detecting the observed outcome [8]. i.e., the maximum likelihood tries to find the estimated coefficients $(\hat{\beta})$ such that feeding these estimates into the model for Pr(X), returns a number approximately to one for all the observations where K = 1 (success), and a number near to zero when K = 0 (failure) [7]. The log-likelihood can be written as:

$$\ell\left(\beta\right)\ =\ \sum_{i=1}^{N} y_i \beta^T x_i - log\left(1\ +\ e^{\beta^T x_i}\right)$$

Here N are the observations and y_i is the response variable (for a two-class case a 0/1 response, usually). For maximizing the log-likelihood, its derivative is set to zero [5]:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i\left(y_i - p(x_i;\beta)\right) = 0$$

To solve the above equation, the Newton-Raphson algorithm is used, which requires the Hessian matrix (the second derivative) [5]. The mathematical details for solving the second derivative through the Newton-Raphson algorithm is beyond of the scope of this report; for more information refer to Han *et al.*, [5]

C. Gradient Boosting Machine

In machine learning, boosting is a method for combining many models into ensembles. Boosting is often used when building ensembles of decision trees. In addition to the initial estimator, each additional estimator depends on the previous one. Due to the sequential stages followed in boosting, and the addition of ensemble members to calculate the overall ensemble prediction, it is also known as stagewise additive modelling.

A gradient boosting algorithm is a type of boosting algorithm. In combination with prior models, it is based on the principle that the best model will minimize overall prediction error. The main idea behind this next model is to set the target outcomes so as to minimize the error. Gradient boosting derives its name from the fact that target outcomes are determined by the degree to which the error deviates from the initial prediction. It creates a new model for each training case that minimizes prediction error.

One widely used machine learning method is XGBoost, a modelling procedure and Python package. The gradient boosting procedure is implemented with XGBoost as an additive stage-by-stage model. In order to build a decision tree ensemble, XGBoost makes use of the gradient descent principle to train new decision trees using the derivative of a loss function. A decision tree can be trained using error gradients instead of node impurity. As XGBoost trains new trees, the aim is to decrease error in ensemble predictions. By adjusting the learning rate hyperparameter, the amount of progress in that direction can be controlled. [1], [2]

In the traditional sequential gradient boosting technique, the process that takes the most time is the split finding process which uses the greedy algorithm. Though the greedy algorithm is fast with small data sets, for very large data sets, the process becomes extremely slow. This is because, the entire data set is linearly scanned, and for each unique value in the data, a tree is built first without considering the effect of the split until the next iteration. It is a highly optimised and well-engineered parallelism in XGBoost that makes the process 10 times faster compared to the gradient boosting technique. The algorithm for gradient boosting is explained below:

1) Input: $x_i, y_{i=1}^n$ and a differential loss function $L(y_i, F(x))$
   Here x_i are the independent features and y_i is the dependent feature.

Given the predicted probability, the log(likelihood) of the data needs to be calculated to find the loss function:

$\sum_{i=1}^{N}[y_i * log(p) + (1 - y_i * log(1-p)]$

Where p is the predicted probability, and y are the observed values.

In order to use the log(likelihood) as a loss function where smaller values represent a better fitting model, the log(likelihood) needs to be multiplied by -1. Thus,

$-[observed * log(p) + (1 - observed) * log(1-p)]$

By simplifying the above equation:

$Lossfunction = -observed * log(odds) + log(1 + e^{log(odds)})$

Both input requirements are ready.

2) Step 1: Initialize the model with a constant value.

$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma)$

This equation is used to find the initial prediction.

$L(y_i, \gamma) =$ Loss Function

y_i refers to the observed values, Gamma refers to the log(odds) values. The argmin over gamma means that is necessary to find a log(odds) value that minimizes $\sum_{i=1}^{n} L(y_i, \gamma)$. $F_0(x)$ is the initial leaf. Hence a leaf has been created that predicts the log(odds) which is nothing but the constant value required.

3) Step 2: for m = 1 to M

a) Calculate the Pseudo Residuals.

$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

for $i = 1, \ldots, n$.

$-\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$ is the derivative of the loss function with respect to the predicted log(odds).

Hence by taking the derivative

$Observed - \frac{e^{log(odds)}}{1+e^{log(odds)}}$

Thus, pseudo residuals can be seen as observed probability minus the predicted probability.
$Residuals = Observed - Predicted$
$F(x) = F_{m-1}(x)$ say to plug in the most recent predicted log(odds). Compute the pseudo residuals for each sample, $r_{i,m}$ where *i* is the sample number and *m* is the tree that is being built.

b) Fit the regression tree to the r_{im} values and create terminal regions R_{jm} for j = 1 to J_m.
Next, a regression tree will be created using the independent variable to predict the residuals and find the terminal regions.

c) For $j = 1$ to J_m compute output values for new tree.

$\gamma_{jm} = \underset{\gamma}{\arg\min} \sum_{x_i \varepsilon R_{ij}} L(y_i, F_{(m-1)}(x_i) + \gamma)$

$j = 1$ to J_m tells that for each leaf in the new tree, compute an output value gamma j, m.
The output value for each leaf is the value for gamma that minimizes the summation.
Assuming that the data set has three rows, the x_i in R_{ij} means that since only the first row of data x_1 goes to leaf R_{11}, then only x_1 is used to calculate the output value for R_{11} and since more than two samples x_2, x_3 go to the leaf R_{21}, then only x_2, x_3 are used to calculate the output values R_{21}.

$\gamma_{1,1} = \underset{\gamma}{\arg\min} - y_1 * [F_{m-1}(X_1) + \gamma] + log(1 + e^{F_{m-1}(X_1)+\gamma})$

Approximate the loss function with a second-order Taylor polynomial and then take derivative w.r.t gamma, and simplify the equation to get a generalized equation of gamma.

$\gamma = \frac{\sum residuals_i}{\sum(p_i * (1-p_i))}$

Therefore, gamma can be defined as the sum of residuals divided by the sum of $p(1-p)$ for each sample in the leaf. Calculate output values for each leaf in the tree.
The algorithm is trying to find the value for gamma that when added to the most recent predicted log(odds) minimizes the Loss Function.

d) Update the model.

$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_m I(x \varepsilon R_{jm})$.

A new prediction is made for each sample. The new prediction will be called $F_1(x)$ and so on. $\sum_{j=1}^{J_m} \gamma_{jm} I(x \varepsilon R_{jm})$ is the output values from the previous tree. ν is the learning rate.
$F_1(x)$ is used to make a new a new prediction for each sample. m will be 2 and the process will be repeated until M is reached.

4) Step 3: Output $F_M(x)$. [3]

III. Results

The logistic regression classifier was considered to be an acceptable model in the heart disease data set since the response variable is dichotomous. According to Stoltzfus (2011), for a binary event, logistic regression is an often method of choice. One advantage of this model is that the observations are not required to have a normal distribution.

TABLE I
PERFORMANCE METRICS FOR THE LOGISTIC REGRESSION AND XGBOOST CLASSIFIERS

Metric	Logistic Regression	XGBoost
Accuracy	0.83	0.87
Precision	0.83	0.87
Recall	0.83	0.87
F1-score	0.83	0.86
TN	70	70
FP	18	18
FN	12	6
TP	78	84

For the logistic regression model, the accuracy of the training and test sets are 0.86 and 0.83, respectively. Overfitting of the model is not implied, considering that both accuracy values are similar. A significantly greater training accuracy over test accuracy could indicate overfitting of the classifier. Table 1 presents the outcomes of the confusion matrix. The logistic regression correctly classified 78 and 70 observations with $K = 1$ (True positives, TP) and $K = 0$ (True negatives, TN), respectively. Twelve subjects with heart disease were incorrectly classified as not having it (False negatives, FN). In contrast, eighteen were classified as having heart illness (False positives, FP). The weighted precision is 0.83, which suggests a low FP rate. The empirical probability of correctly predicting heart disease is 0.83 (sensitivity or recall). Since the weighted average of precision and sensitivity have the same value, the F1 score is also 0.83.

The logistic regression was conducted under several assumptions. First, independence of errors (no duplicate responses), linearity in the logit for continuous covariates, and absence of multicollinearity. A disadvantage is that this model is susceptible to outliers [8]. Considering this, the test accuracy might be improved if observations reporting cholesterol levels equal to 0 were excluded from the model.

Comparatively to other algorithms, XGBoost is a simple and efficient algorithm that delivers excellent performance and accuracy. There might be several reasons for this. Regularization is inbuilt into XGBoost, preventing overfitting. Parallel processing allows XGBoost to be much faster than the traditional Gradient Boosting algorithm. The algorithm can deal with missing values by default. XGBoost allows users to run cross-validation at every stage of the boosting process. XGBoost splits up to the max depth parameter and then starts pruning the tree backward and removes splits beyond which there is no gain. To calculate the scores for each node and leaf in the tree quickly, XGBoost stores the first and second order derivatives (gradients and hessians) in the cache memory. As part of the parallel learning process, XGBoost divides the dataset into multiple Compressed Sparse Column (CSC) blocks for distribution among many cores.

Using the gradient boosting model, with a threshold of 0.5, a testing accuracy of 87% accuracy was obtained. Overfitting appears to be less probable. From the confusion matrix, it can be noted that the number of true positives and true negatives is greater than the number of false positives and negatives.

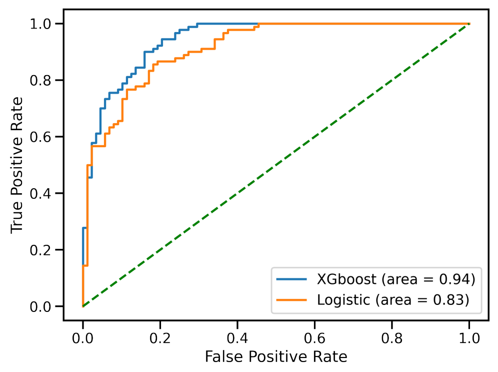


Fig. 1. Receiver Operating Characteristic (ROC) plot for Logistic regression and XGBoost classifiers

The ROC plot (Fig. 1) shows a curve near the top-left corner, indicating good performance for both models. According to Mandrekar (2010) [10], ROC curves above the diagonal have a relatively good discriminating ability to diagnose patients with and without the disease/condition. In contrast, the closer the curve is to the diagonal (when sensitivity = 1 - specificity), the less accurate is the classifier. i.e., the model has no discriminatory ability. The area under the curve (AUC) for Logistic Regression is 0.83 and for XGBoost is 0.94. This means that for a randomly chosen pair of subjects with and without heart disease, the ill subject has an 83% and 94% chance of being correctly diagnosed. Since the AUC for XGBoost is greater than 0.85, it is appropriate to imply that the classifier exhibits very good performance.

IV. CONCLUSION

The two machine learning models were applied successfully. From the results, it can be seen that the gradient boosting algorithm implemented using XGBoost gives the best accuracy compared to the Logistic Regression Model. It is explained in detail how each model works and the mathematical intuition behind each. From this, it can be said that since the gradient boosting model works on the idea that with each iteration the error must be reduced, it is giving a high accuracy in predicting whether the person is likely to have heart disease. Even though that's the case, wouldn't be appropriate to establish that this model is the only one that can be used; as the reasons behind other models having less accuracy could be due to the tuning of the parameters, requirement of more data and data preprocessing. The classifiers should be tested in a real world situation (physicians supporting a diagnosis). Only then the correct performance of the models can be judged. Nonetheless, for the scope of this paper, it is appropriate to establish that the XGBoost classifier is the best.

References

[1] Klosterman, Stephen. *Data Science Projects with Python*. Packt Publishing, Limited, 2019.

[2] Chen, Tianqi and Guestrin, Carlos, "XGBoost: A Scalable Tree Boosting System", Association for Computing Machinery, 2016

[3] Gradient boosting. [Online]. Available: $https://en.wikipedia.org/wiki/Gradient_boosting$

[4] Aggarwal, Charu C., "Data Mining". 2015th ed., Springer International Publishing.

[5] Han, Jiawei, et al. *"Data Mining"*. Elsevier Science Technology, 2011.

[6] Alexandropoulos, Stamatios-Aggelos N, et al. *"Data Preprocessing in Predictive Data Mining". Knowledge Engineering Review*, vol. 34, 2019, pp. Knowledge engineering review, 2019, Vol.34.

[7] James, Gareth, et al. *"An Introduction to Statistical Learning"*. Vol. 103, Springer, 2013.

[8] Stoltzfus, Jill C. "Logistic Regression: A Brief Primer." *Academic Emergency Medicine*, vol. 18, no. 10, 2011, pp. 1099–1104.

[9] Tolles, Juliana, and William J Meurer. "Logistic Regression: Relating Patient Characteristics to Outcomes." *JAMA : the Journal of the American Medical Association*, vol. 316, no. 5, 2016, pp. 533–534.

[10] Mandrekar, Jayawant N. "Receiver Operating Characteristic Curve in Diagnostic Test Assessment." *Journal of Thoracic Oncology*, vol. 5, no. 9, 2010, pp. 1315–1316.

[11] World Health Organisation, "The top 10 Causes of death", 2009, https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

[12] Dana Kauffman, "Cardiovascular Disease Burden, Deaths are Rising Around the World." *American College of Cardiology*, 2020

A Comparison of the Performance of Machine Learning Models in Predicting Heart Disease

Group 14

Department of Computer Science and Communication
Lancaster University

January 13, 2022

Overview

Heart disease worldwide and relevance of classifiers

- Heart disease was responsible for 16 million deaths globally in 2019.
- Early detection given previous conditions and other factors is needed more than ever.
- Given several factors such as age and pre-existing conditions, classification machine learning algorithms can be of great help in predicting possible heart disease

Heart data set, application to a real matter

- A heart data set consisting of 918 observations was used to test classifiers to determine a patient's heart disease status.
- The data contains eleven covariates and one binary response variable.
- Features:
  - Numerical: age, blood pressure, serum cholesterol, fasting blood sugar, maximum heart rate, and old peak.
  - Nominal: sex, chest pain type, and exercise-induced angina.
  - Ordinal: resting electrocardiogram and slope of the peak exercise ST segment

Data pre-processing

- Encoding categorical variables:
  - Ordinal encoding
  - Dummy variable encoding
- Correction of inconsistencies:
  - Cholesterol attribute presented zero values.
  - Zeros were replaced by the median.
- Standardisation of numerical variables.
- Detection of outliers: Inputs outside the [-3, 3] range were dropped.

Logistic Regression (for binary classification)

- Logistic regression models the probability of the response variable to belong to a K class via linear functions in x.
- To prevent probabilities outside the $[0, 1]$ ranges, the model uses the logistic function.
- For a dependent variable y on a set of independent variables (covariates) $\mathbf{x}$, the objective is to find the logistic regression function $\boldsymbol{p(x)}$ that returns accurate responses of $\boldsymbol{p(x_i)}$ for each observation $i = 1, .., n$.
- If the dependent variable is dichotomous, the response can have only two values, often 0 and 1.

Logistic Regression (for binary classification)

- The logistic regression function is built estimating β coefficients.
- A linear function called **logit**, is needed for estimating the coefficients:

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + ... + \beta_r x_r$$

  Where r corresponds to the number of covariates.
- The logistic regression function $p(\boldsymbol{x})$ is the sigmoid function of $f(\boldsymbol{x})$, so:

$$p(\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}},$$

$$log(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}) = f(\mathbf{x})$$

  For a value of $\boldsymbol{x}$, if $p(x) = 1$, then, $1 - p(x) = 0$.

Estimation of the β_r

- Estimation is done through training of the classifier.
- Coefficients are obtained for all the observations maximizing the log-likelihood function:

$$l = \sum_{i=1}^{N} (y_i log(p(\boldsymbol{x_i})) + (1 - y_i) log(1 - p(\boldsymbol{x_i})))$$

If $y_i = 0$, $l = log(1 - p(\boldsymbol{x_i}))$. When $y_i = 1$, $l = log(p(\boldsymbol{x_i}))$.

- When the β are determined, the model is ready to predict any $p(x_i)$ for any value of $\boldsymbol{x}$. A threshold of 0.5 is usually employed. Therefore, if $p(x_i) > 0.5$ then the output is 1, otherwise is 0.

Logistic Regression Results (estimated β)

Table: Estimated coefficients for the logistic regression model

Covariate	**Category**	β_i
Intercept		0.71
Age		1.69
Sex	Female	-0.24
Sex	Male	0.95
ChestPainType	Asymptomatic	1.39
ChestPainType	Atypical Angina	-0.65
ChestPainType	Non-Anginal Pain	-0.12
ChestPainType	Typical Angina	0.08
RestingBp		3.04
Cholesterol		7.63
FastingBS		1.05
RestingECG		0.08
MaxHR		-6.00
ExerciseAngina	No	-0.03
ExerciseAngina	Yes	0.75
Oldpeak		0.33
STSlope		-1.61

Gradient Boosting Machines

- Gradient Boosting is a type of boosting algorithm combining many models into ensembles.
- It is based on the principle that the best model will minimize overall prediction error.
- It creates a new model for each training case that minimizes prediction error.

XGBoost algorithm

- Input: $x_i, y_{i=1}^n$ and a differential loss function $L(y_i, F(x))$ Here x_i are the independent features and y_i is the dependent feature.

- Given the predicted probability, the log(likelihood) of the data needs to be calculated to find the loss function:

$$\sum_{i=1}^{N}[y_i * log(p) + (1 - y_i * log(1 - p)]$$

- Where p is the predicted probability, and y are the observed values.

- Simplifying the above equation:

$$\text{Loss function} = -observed * log(odds) + log(1 + e^{log(odds)})$$

- Step 1: Initialize the model with a constant value.

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

This equation is used to find the initial prediction.

$$L(y_i, \gamma) = \text{Loss Function}$$

y_i refers to the observed values, γ refers to the log(odds) values. The argmin over gamma means that is necessary to find a log(odds) value that minimizes $\sum_{i=1}^{n} L(y_i, \gamma)$. $F_0(x)$ is the initial leaf. Hence a leaf has been created that predicts the log(odds) which is nothing but the constant value required.

- Step 2: for m = 1 to M calculate the Pseudo Residuals.

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

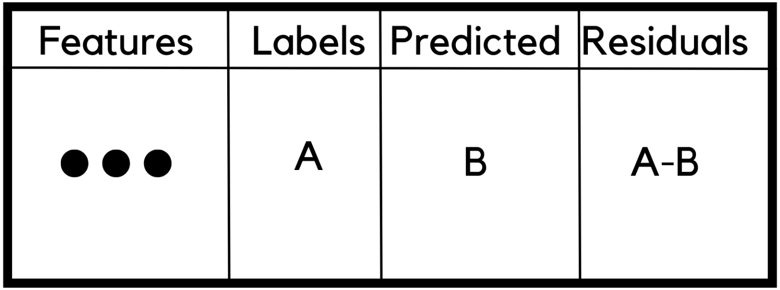
for $i = 1, \ldots, n$.

$$-\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$$

Is the derivative of the loss function with respect to the predicted log(odds). Hence, by taking the derivative:

$$Observed - \frac{e^{log(odds)}}{1 + e^{log(odds)}}$$

Thus, pseudo residuals can be seen as observed probability minus the predicted probability. $Residuals = Observed - Predicted$. $F(x) = F_{m-1}(x)$ say to plug in the most recent predicted log(odds). Compute the pseudo residuals for each sample, $r_{i,m}$ where i is the sample number and m is the tree that is being built.



Features	Labels	Predicted	Residuals
●●●	A	B	A-B

Figure: The figure shows the x_i, y_i, and how the residuals r_{im} are calculates.

- Fit the regression tree to the r_{im} values and create terminal regions R_{jm} for j $= 1$ to J_m.
- Next, a regression tree will be created using the independent variable to predict the residuals and find the terminal regions.

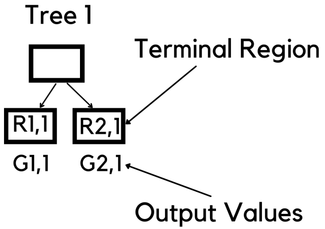


Figure: Figures show how a decision tree is formed and explains the various terms related to the equations above.

- For $j = 1$ to J_m compute output values for new tree.

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \varepsilon R_{ij}} L(y_i, F_{(m-1)}(x_i) + \gamma)$$

$j = 1$ to J_m tells that for each leaf in the new tree, compute an output value gamma j, m. The output value for each leaf is the value for gamma that minimizes the summation.

- Approximate the loss function with a second-order Taylor polynomial and then take derivative w.r.t gamma, and simplify the equation to get a generalized equation of gamma.

$$\gamma = \frac{\sum residuals_i}{\sum(p_i * (1 - p_i))}$$

- Therefore, gamma can be defined as the sum of residuals divided by the sum of $p(1-p)$ for each sample in the leaf. Calculate output values for each leaf in the tree.

- Aim, to find the value for gamma that when added to the most recent predicted log(odds) minimizes the Loss Function.

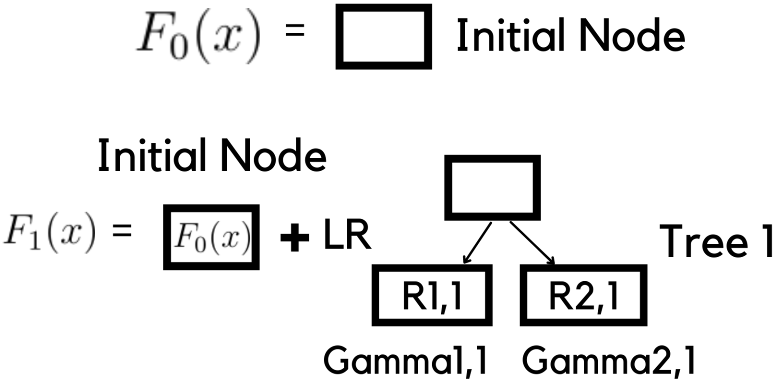


Figure: The figure tries to visualize the first iteration i.e m $= 1$

- Update the model.

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_m I(x \varepsilon R_{jm})$$

- A new prediction is made for each sample. The new prediction will be called $F_1(x)$ and so on. $\sum_{j=1}^{J_m} \gamma_{jm} I(x \varepsilon R_{jm})$ is the output values from the previous tree. ν is the learning rate.

- $F_1(x)$ is used to make a new a new prediction for each sample. m will be 2 and the process will be repeated until M is reached.

- Step 3: Output $F_M(x)$.

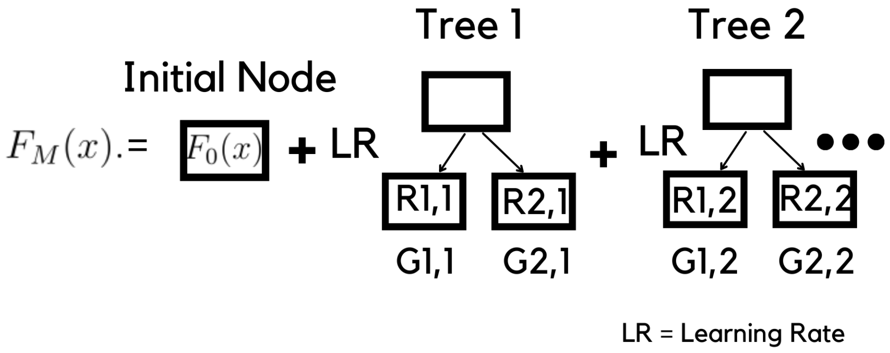


Figure: The figure represents how the entire model will look like for m = M

Gradient boosting was implemented using XGBoost. XGBoost gives a higher performance than other algorithms. And this might be the various advantages it offers.

- Inbuilt regularization
- Parallel processing
- Deals with missing values
- Cache optimisation
- Distributed computing

Results

Results obtained from the implementation of both models.

Table: Performance metrics for the Logistic Regression and XGBoost classifiers

Metric	Logistic Regression	XGBoost
Accuracy	0.83	0.87
Precision	0.83	0.87
Recall	0.83	0.87
F1-score	0.83	0.86
TN	70	70
FP	18	18
FN	12	6
TP	78	84

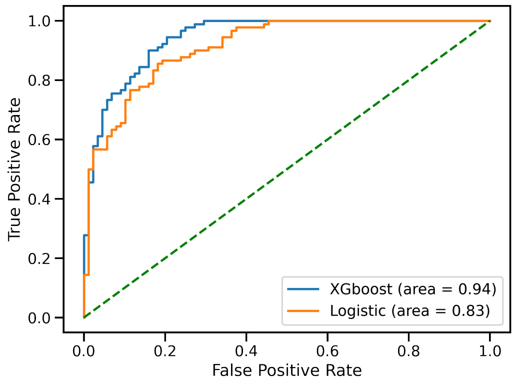


Figure: Receiver Operating Characteristic (ROC) plot for Logistic regression and XGBoost classifiers

Conclusions

- The two machine learning models were applied successfully.
- It can be seen that the gradient boosting algorithm implemented using XGBoost gives the best accuracy compared to the Logistic Regression Model.
- Even though XGBoost algorithm has better performance metrics, wouldn't be appropriate to establish that this model is the only one that can be used.
- Reasons behind other models having less accuracy could be due to the tuning of the parameters, requirement of more data and data preprocessing.
- The mathematical intuition behind each algorithms has been successfully explained.

Feedback comments

A very well written and explained report. Your discussions of choices in the data cleaning, and the important aspects of the models were very good. Figures and tables were all well presented, it may have been nice to dig a little further in to the conclusions in the context of the real dataset - what variables were most important etc?

Grading

Group project rubric. Quite general. Awards points at sectional level.

Grading is from 0-5 in four categories, as explained in student-facing outline. 5 = Outstanding, 4 = Excellent, 3 = Good, 2 = Satisfactory, 1 = Poor, 0 = No Credit.

Abstract/Title

No Credit	Poor	Satisfactory	Good	Excellent	[✓ Outstanding]

Introduction

No Credit	Poor	Satisfactory	[✓ Good]	Excellent	Outstanding

Methods

No Credit	Poor	Satisfactory	Good	[✓ Excellent]	Outstanding

Detailed discussion of the learning process and logistic regression. Just make sure that all notation has been explicitly defined.

Results

No Credit	Poor	Satisfactory	[✓ Good]	Excellent	Outstanding

Good work. It would be nice to understand the estimated impact of individual variables a little more since we have an interesting real dataset.

Discussion

No Credit	Poor	Satisfactory	[✓ Good]	Excellent	Outstanding

Presentation

No Credit	Poor	Satisfactory	Good	[✓ Excellent]	Outstanding

Well presented.

Detail

No Credit	Poor	Satisfactory	Good	[✓ Excellent]	Outstanding

Very good. Well justified decisions throughout. May need a little more detail on trees to underpin the XGBoost discussion.

SPAG

No Credit	Poor	Satisfactory	Good	[✓ Excellent]	Outstanding

Very good. Make sure all mathematical notation is typeset within $'s so it is italicised.

References

No Credit.	Too Few/Not appropriate	[✓ Good]	Excellent

74.42 / 100.00