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

1. Heart disease worldwide and relevance of classifiers
2. Heart data set, application to a real matter
3. Data preparation for the implementation of the proposed models
4. Implemented Classifiers
   - Logistic Regression
   - XGBoost
5. Performance evaluation
6. Conclusions

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.

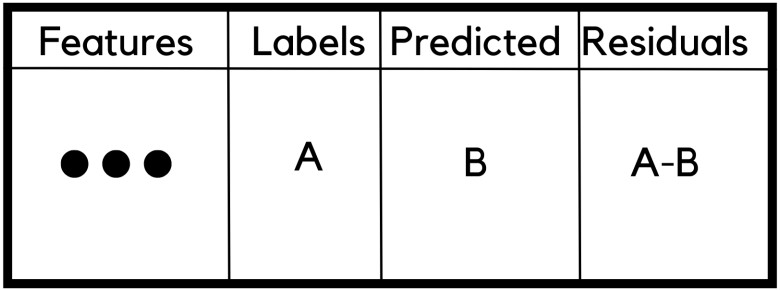

Figure: The figure shows the x_i, y_i, and how the residuals r_{im} are calculates.

- 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.

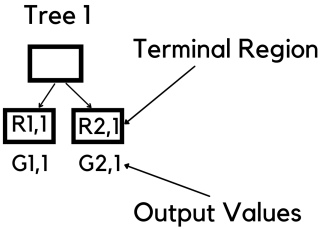

Figure: Figures show how a decision tree is formed and explains the various terms related to the equations above.

- 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.

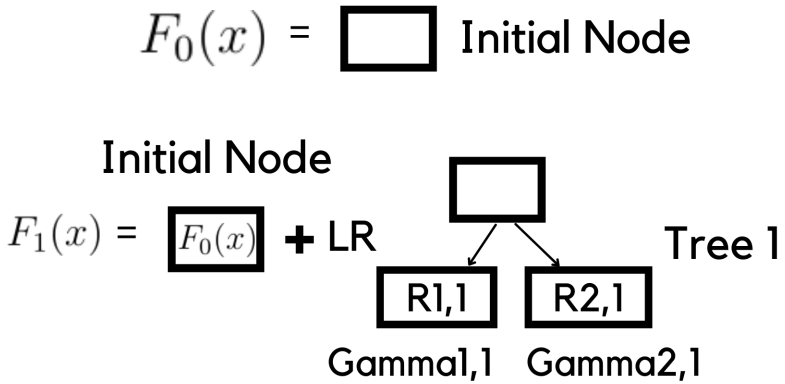

Figure: The figure tries to visualize the first iteration i.e m $= 1$

- 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 $F_M(x)$.

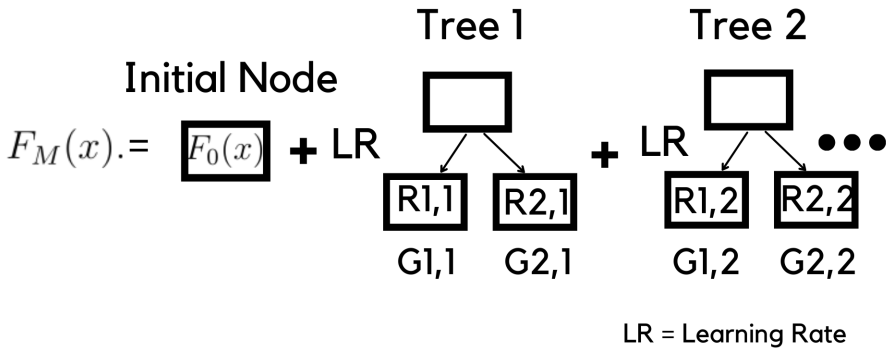

Figure: The figure represents how the entire model will look like for m = M

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

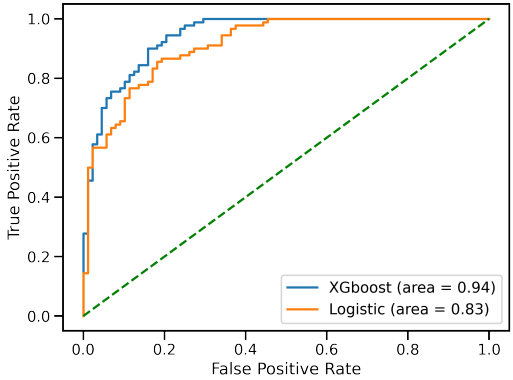

Figure: Receiver Operating Characteristic (ROC) plot for Logistic regression and XGBoost classifiers

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.