The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project 7

Predicting heart disease using Binomial Logistic Regression

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Abstract

This Project is based on the use of Binomial logistic regression algorithm which is one of the well-known types of general linear model algorithms. It is the most widely implemented algorithm to make relationship between binary outcome variable and predictor or independent variable, model building, casual inference, and hypothesis testing.

The data used in this project is called “heart.csv”, which is a real-world dataset from the UCI machine learning repository. The dataset includes a total of 270 observations of a number of patients with potential heart diseases and a number of variables.

In this project we will use this data and perform an analysis for the same using Binomial logistic regression algorithm estimation and find which independent variables (such as sex, age, ECG, etc.) can be good predictors of potential heart diseases in patients. We will prepare and handle data, train the model and then evaluate and improve the performance.

Introduction

Binomial Logistic Regression

In this project the model will be based on Binomial regression which is a type of general linear model used to handle data or variables which are binomially distributed. As the name suggests, the model will take variables which are binary in nature. For example, in this project, the model used factors which are first changed to variables which are coded as 0 or 1, which will simply indicate the presence or absence of an outcome.

In a binomial logistic regression, the handles one or more predictors to model the probability. These variables can be different in nature, that is they can be continuous, categorical or can be a combination of both. The model will observe the effects of these variables with their probability being a 1, which would be termed as a success or 0 being a failure. In the logistic regression model, the coefficients indicate how the log odds of the response variable change for a one-unit increase in the predictor variable.

This logistic regression is widely used in science related fields such as epidemiology, biostatistics, or other areas where the desired outcome is often binary in nature and does not demand fixed values or a range in result. Some good examples of the use of binary logistic regression model would be presence or absence of genetic conditions, disease or abnormality, death or survival or a success or failure in any scenario.

Maximum likelihood Algorithm

This is a statistical tool used to estimate the probability distribution of observed data. This algorithm focuses on maximizing likelihood by minimizing the deviance in a model which can also be defined as maximizing log-likelihood to estimate the parameters is the same as minimizing the model deviance. This algorithm implemented in a model assumes that the observations are independent and have the same distribution, with the aim to find values of the parameter for which the observed data have the maximum likelihood or in other words how likely the data is for a given parameters value in a model.

The algorithm will also involve the finding the partial derivatives of likelihood function for each parameter while setting them equal to zero, which will result in a system of equations which can be solved to find the maximum likelihood of the parameters.

Even though the algorithm is popular for its efficient and consistent properties and is widely used in a variety of statistical problems such as logistic regression, linear regression, etc. It is not a cure for all the situations. For example, the model is sensitive to the first value of the parameter and may not cover the overall maximum of likelihood function. At the same time there are no alternate criteria on which the model works.

Problem Description

Heart disease is one of the major causes of death for some of the major countries like India, USA etc. Medical research has often proven that there are certain factors from a patient’s lifestyle, diet and experiences which could be good predictors of someone having a heart disease or potential of developing one. For Example, Factors from lifestyle such as, exercise, cholesterol, and blood sugar levels etc.

Therefore, even though heart related diseases (such as heart attacks or cardiac arrest) happen randomly in an individual’s life. It seems that there are factors which can be taken into consideration to determine if an individual is likely going to suffer from one soon. Creating a model based on machine learning could provide an effective strategy to highlight factors which are most prevalent in determining the future presence of heart disease.

The goal of this analysis is to find out variables or factors which could be good predictors of a potential heart disease in a person. An effective model could be a deep insight and give direction to prevent such a condition.

Objectives:

The primary goal of this report is to:

* To comprehensively introduce Binomial logistic regression algorithm, explaining its mathematical foundations.
* To introduce Maximum likelihood estimation algorithm.
* Improving the model using different methods (Example lasso technique) .
* Using visualizing tool to understand the threshold and other aspect of data.
* Outline the method involved in building and training the model.
* Propose recommendations for future research and development in this field.

Method

This project will be using the dataset “heart.csv.” The primary source of literature used is “Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)”.

Listed below are the steps taken in the report:

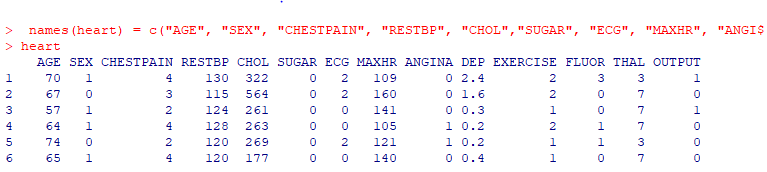
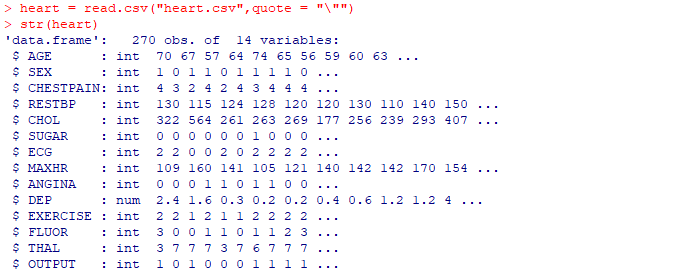
* Step 1 – Collecting data.
* Step 2 – Exploring and preparing the data.
* Step 3 – Model Deviance
* Step 4 – Test set performance
* Step 5 – Regularization with the lasso
* Step 6 – Classification metric
* Step 7 – Visual Representation

Steps taken:

Collecting data.

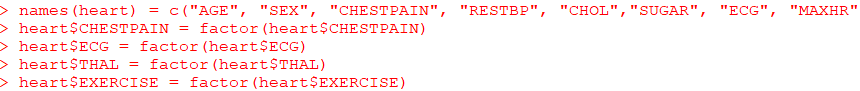
The data is stored in a file named “heart.csv” in the documents. We will use the read.csv command to load the data which contains 270 observations.

We will also rename the columns as per the instructions.



Exploring and preparing the data.

One important criterion of the binomial logistic regression model are that the features should be numerical variables and they are coded in numbers. In this data, we see that the features like “Chest pain”, “Thal”, “ECG” and “Exercise” are all categorical variables. We use the factor () command to change the variable into categorical variable which will be changed to binary valued indicator variable.



We will not make any change to the “Output” variable as it already exists in the binary form (1 or 0).

This prepares our data frame for the model. We will split the data frame into two parts, that is training and testing set. We would set a seed value to create randomization and we will make a split of 85 to 15 for the two data sets. Which means 85% of data is fed into the model and will be the training set to train the model. 15% would be the test set to test the model.

We use the glm() function to train our logistic regression model. The model formula states that we identify output variable and all the features we use. The “family parameter suggest the type of regression we want to perform.

We use the summary command to see the model we trained.

A picture containing text, font

Description automatically generated

We see that the median residual value is -0.13 with a maximum value as 2.88 and a minimum value as -2.51.

We see that the three valued Thal produced two binary features named Thal6 and Thal7. All the regression coefficients have their respective z-values. A high Z value will suggest that the corresponding feature is more likely to have an impact on the output variable.

A screenshot of a computer

Description automatically generated with medium confidence

As the models are trained with maximum likelihood criteria, we use standard normal distribution to check significance of a coefficient. For example, in this case we use z value of FLUOR we set the lower tail parameter to T to test negative coefficients, this will return us the value of p for FLUOR.



We see from the summary that the “Chestpain4”,” Thal7” ,”SEX” and” FLUOR” are the strongest predictors of a potential heart disease as they have the high p value. We also see a number of other features with high p value. This suggests that they are not good predictors of heart diseases. This may sound counterintuitive, but what we should interpret from the table is that in presence of all these features with high p-value factors which truly affect heart disease are unable to impact in the model. This is also due to the collinearity among these features. For example, we use the feature Sugar, which has a negative z value, but we know Sugar is supposed to have a positive correlation as more blood Sugar levels, the higher there is a chance of heart diseases. The phenomenon is happening due to the presence of other factors with high p value making Sugar insignificant in the model.

We can check this by creating a model only with the SUGAR variable. We see a positive regression coefficient with a low p value, which confirms that the other features are collinear nature.

A screenshot of a computer

Description automatically generated with medium confidence

Model Deviance

This is an important concept of Logistic regression models. In simple terms we can say that the deviance of a dataset is the sum of all the observed deviance. The deviance of an observation is calculated by multiplying the log likelihood of that model with -2.

Mathematically, the deviance residual is analogous to the residual of linear regression. It also means minimizing the sum of squared deviance residual is the same as maximizing the log likelihood.

To do the same in our model we will use the log\_likelihood feature to compute vector of log likelihood for all observations using the predicted model probabilities and target labels. We use the dataset\_log\_likelihood() to sum up to produce log likelihood of a data set.

Now we compute two analogous functions which are deviance() and dataset\_deviance(). Thses produce observation deviance in form of a vector and the latter add all these up.

We can now use predict() function to compute deviance in the training data. This will return probabilities on a logit scale. We will be specifying the value of response for the type of parameters.

To check the function we will compute the residual deviance of heart\_model.

A screen shot of a computer code

Description automatically generated with low confidence

Another way to check the logistic regression model is by calculating the difference between deviance of the model and deviance of a null model. Null deviance can be defined as a deviance of a model which is trained without any features. The null model will predicts class 1 through a constant probability due to absence of any features.

A picture containing text, font, screenshot

Description automatically generated

We note that we produced the value from the summary as seen above. The null deviance and the residual deviance are comparable to the True Sum of Square (TSS) and Residual Sum of Squares (RSS). When the discrepancy between the two is significant, it can be associated to the concept of residual sum of squares in linear regression, which is responsible for observed variance in the output variable.

We are defining pseudo R2 for our model using the same equation. This shows that the model is roughly 56 percent of the null deviance. This is not high as the features are not high enough to make accurate predictions. In some cases when residual deviance exceeds the null deviance, we might get an R square more than 1. In such case we discard the model and try alternative methods such as feature selection or trying a new model.

A picture containing text, font, screenshot

Description automatically generated

The absence of p value indicates that there is no test generated by R. We observe that the difference between the null and residual deviance is Chi squared distributed. We can compute p-value for the same.

First, we calculate difference between residual and null deviance. We compute the degree of degree of freedom by subtracting the degree of freedom between our model and null model which will be total number of observations in our model minus 1. For residual deviance we calculate a number of regression coefficient by subtracting this number from total number of observations.

We then use pchisq() function to get p value, we will set the lower.tail parameter as False as we are making a upper tail calculation.

A close-up of a computer code

Description automatically generated with low confidence

The p value generated is small, hence the prediction is better than the average. We will write a function to define a function to calculate vector of deviance residual.

We use the summary function to obtain a table.

A picture containing text, font, screenshot

Description automatically generated

Test set performance

To test the performance of the model we will perform a binary classification by setting a limit or threshold which we will do with both our training and testing set which we will later compare to measure classification accuracy.

A picture containing text, font, screenshot

Description automatically generated

We observe a good starting point as the accuracy for both the testing and training set look close to eachother. We also see that several features are not significant and there is collinearity present. Also, the pseudo-R square shows that we did not explain deviation in the model. Hence removing some of the features with specific variable selection is needed to make the model effective.

Regularization with the lasso

Due to high collinearity among features, it would be a better idea to remove some of the features. We will do this using the lasso to the dataset. We will train the model with glmnet() and the use cv.glmnet(0 to estimate suitable value for lambda

A picture containing text, font, screenshot

Description automatically generated

We see that a number of features are now removed as their coefficient is absent. We see that the accuracy for the training datasets is now slightly improved, but the accuracy of the testing datasets is now slightly reduced, this might be because of the low data quantity (only 270 observations). As lasso improved accuracy for training set which have more observations but failed to do the same for testing set which has less observations.

A screenshot of a computer code

Description automatically generated with low confidence

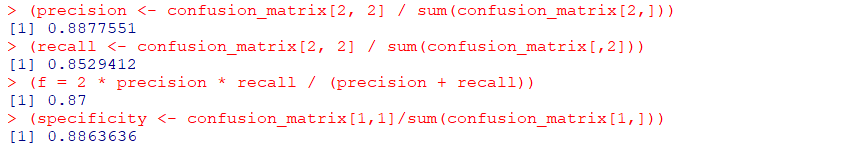
Classification metric

We compute a binary confusion metric. We use a technique of enclosing assignment statements in brackets to assign the outcome of an expression to a variable and show assigned. The effective measure of a condition by a model (in case of a medical scenario) is known as sensitivity. Which is comparable to true positives. The opposite of the same would-be specificity, which can be false negatives in this case. In our case the specificity would be the ability of our model to reject false negatives represented as class1.

A picture containing text, font, screenshot, line

Description automatically generated

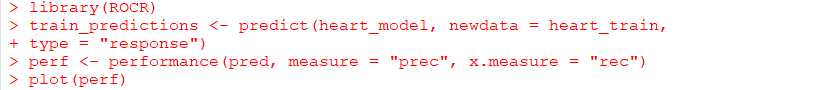
We can compute that by using the following function:

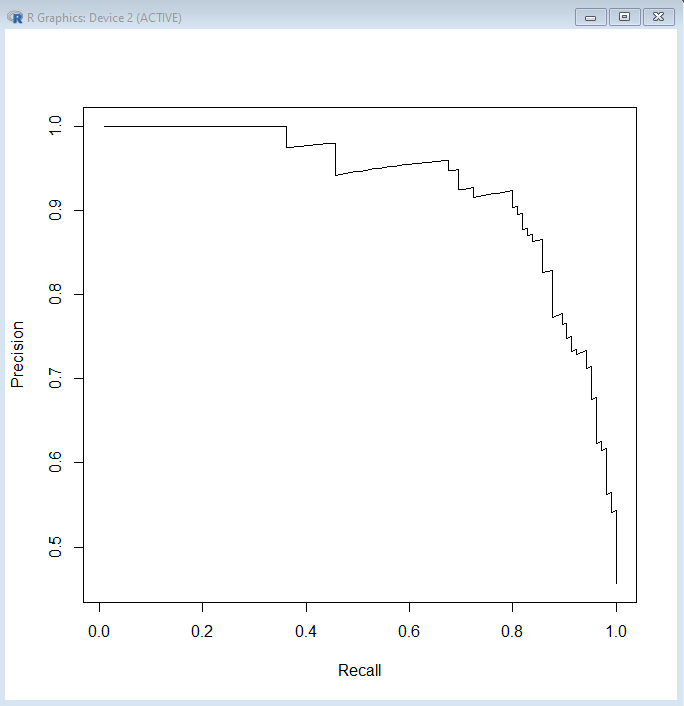


We also realize the importance of setting a threshold value. As a new threshold value would change all the metric. For example, in this case, it is better for a patient to be misdiagnosed with heart disease than for a patient with heart disease to be deemed healthy. We can make our model more rigid or biased for this situation by lowering the threshold value.

Visual Representation

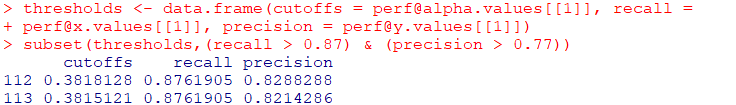
We could visually understand the change in threshold curve by plotting a precision recall curve.





We see in the graph that there is a sharp change in value above 0.8. To fine tune this situation we could see individual graphs. We could do this by making data frame of cutoff and threshold values which should be used to recall change in data with their precision and respective values.

We would want to have a threshold which could return 87%of recall and 77% precision.



We see that a threshold of roughly 0.38 is good to satisfy requirements.

Conclusion

Through this project we conclude that:

* Binomial logistic regression is a good model to predict heart diseases after changing data to binary values.
* Data split evenly in two datasets to build and test the model.
* We use functions to compute model and null deviance.
* Even when the p value of some features is significantly high, they should be removed as they are often collinear and makes the other features with defining correlation insignificant in the model output.
* Both test and training set getting approximately same value in terms of accuracy.
* We use lasso function to remove the features with correlation, which later improves the accuracy for the new training model but not the testing model.
* We plot a precision recall curve for understanding the optimal threshold value which in this case is 0.38.

Limitations and Further improvements

Some limitations of the Binomial logistic regression algorithm are:

* The model is prone to multicollinearity, that is if two or more predictors are collinear in nature , it will give an ineffective coefficient and error.
* It works on the principal that the variables are independent and have homogeneous variance.
* It is often difficult to choose predictors variables.
* If a data has outliers, then the coefficient would be affected as the model is sensitive to outliers.
* The model is based on linear assumption and will give wrong result s if this assumption is violated.
* Like other models this model is prone to overfitting.

Some potential improvements of the Binomial logistic regression algorithm are:

* Nonlinear transformation of the predictor variables to capture complex relationships.
* Using variable selection technique to address the issue of multicollinearity.
* Using nonparametric models to find relation between parameters.
* Using bagging and boosting to improve the model’s accuracy.
* Using lasso or Ridge regression to address overfitting.

Coding

heart = read.csv("heart.csv",quote = "\"")

str(heart)

names(heart) = c("AGE", "SEX", "CHESTPAIN", "RESTBP", "CHOL","SUGAR", "ECG", "MAXHR", "ANGINA", "DEP", "EXERCISE", "FLUOR","THAL", "OUTPUT")

heart$CHESTPAIN = factor(heart$CHESTPAIN)

heart$ECG = factor(heart$ECG)

heart$THAL = factor(heart$THAL)

heart$EXERCISE = factor(heart$EXERCISE)

library(caret)

set.seed(987654)

heart\_sampling\_vector = createDataPartition(heart$OUTPUT, p = 0.85, list = FALSE)

heart\_train = heart[heart\_sampling\_vector,]

heart\_train\_labels = heart$OUTPUT[heart\_sampling\_vector]

heart\_test = heart[-heart\_sampling\_vector,]

heart\_train\_labels = heart$OUTPUT[heart\_sampling\_vector]

heart\_model = glm(OUTPUT ~ ., data = heart\_train, family = binomial("logit"))

summary(heart\_model)

pnorm(3.846 , lower.tail = F) \* 2

heart\_model2 = glm(OUTPUT ~ SUGAR, data = heart\_train, family = binomial("logit"))

summary(heart\_model2)

log\_likelihoods = function(y\_labels, y\_probs) {

y\_a = as.numeric(y\_labels)

y\_p = as.numeric(y\_probs)

y\_a \* log(y\_p) + (1 - y\_a) \* log(1 - y\_p)

}

dataset\_log\_likelihood = function(y\_labels, y\_probs) {

sum(log\_likelihoods(y\_labels, y\_probs))

}

deviances = function(y\_labels, y\_probs) {

-2 \* log\_likelihoods(y\_labels, y\_probs)

}

dataset\_deviance = function(y\_labels, y\_probs) {

sum(deviances(y\_labels, y\_probs))

}

model\_deviance = function(model, data, output\_column) {

y\_labels = data[[output\_column]]

y\_probs = predict(model, newdata = data, type = "response")

dataset\_deviance(y\_labels, y\_probs)

}

model\_deviance(heart\_model, data = heart\_train, output\_column = "OUTPUT")

null\_deviance = function(data, output\_column) {

y\_labels = data[[output\_column]]

y\_probs = mean(data[[output\_column]])

dataset\_deviance(y\_labels, y\_probs)

}

null\_deviance(data = heart\_train, output\_column = "OUTPUT")

model\_pseudo\_r\_squared = function(model, data, output\_column) {

1 - ( model\_deviance(model, data, output\_column) /

null\_deviance(data, output\_column) )

}

model\_pseudo\_r\_squared(heart\_model, data = heart\_train,output\_column = "OUTPUT")

model\_chi\_squared\_p\_value = function(model,

data, output\_column) {

null\_df = nrow(data) - 1

model\_df = nrow(data) - length(model$coefficients)

difference\_df = null\_df - model\_df

null\_deviance = null\_deviance(data, output\_column)

m\_deviance = model\_deviance(model, data, output\_column)

difference\_deviance = null\_deviance - m\_deviance

pchisq(difference\_deviance, difference\_df,lower.tail = F)

}

model\_chi\_squared\_p\_value(heart\_model, data = heart\_train,output\_column = "OUTPUT")

model\_deviance\_residuals = function(model, data, output\_column) {

y\_labels = data[[output\_column]]

y\_probs = predict(model, newdata = data, type = "response")

residual\_sign = sign(y\_labels - y\_probs)

residuals = sqrt(deviances(y\_labels, y\_probs))

residual\_sign \* residuals

}

summary(model\_deviance\_residuals(heart\_model, data = heart\_train, output\_column = "OUTPUT"))

train\_predictions <- predict(heart\_model, newdata = heart\_train,type = "response")

train\_class\_predictions <- as.numeric(train\_predictions > 0.5)

mean(train\_class\_predictions == heart\_train$OUTPUT)

test\_predictions <- predict(heart\_model, newdata = heart\_test,type = "response")

test\_class\_predictions <- as.numeric(test\_predictions > 0.5)

mean(test\_class\_predictions == heart\_test$OUTPUT)

library(glmnet)

heart\_train\_mat <- model.matrix(OUTPUT ~ ., heart\_train)[,-1]

lambdas <- 10 ^ seq(8, -4, length = 250)

heart\_models\_lasso <- glmnet(heart\_train\_mat,heart\_train$OUTPUT, alpha = 1, lambda = lambdas, family = "binomial")

lasso.cv <- cv.glmnet(heart\_train\_mat, heart\_train$OUTPUT,alpha = 1,lambda = lambdas, family = "binomial")

lambda\_lasso <- lasso.cv$lambda.min

lambda\_lasso

predict(heart\_models\_lasso, type = "coefficients", s = lambda\_lasso)

lasso\_train\_predictions <- predict(heart\_models\_lasso,

s = lambda\_lasso, newx = heart\_train\_mat, type = "response")

lasso\_train\_class\_predictions <-

as.numeric(lasso\_train\_predictions > 0.5)

mean(lasso\_train\_class\_predictions == heart\_train$OUTPUT)

heart\_test\_mat <- model.matrix(OUTPUT ~ ., heart\_test)[,-1]

lasso\_test\_predictions <- predict(heart\_models\_lasso,

s = lambda\_lasso, newx = heart\_test\_mat, type = "response")

lasso\_test\_class\_predictions <-

as.numeric(lasso\_test\_predictions > 0.5)

mean(lasso\_test\_class\_predictions == heart\_test$OUTPUT)

(confusion\_matrix <- table(predicted =

train\_class\_predictions, actual = heart\_train$OUTPUT))

(precision <- confusion\_matrix[2, 2] / sum(confusion\_matrix[2,]))

(recall <- confusion\_matrix[2, 2] / sum(confusion\_matrix[,2]))

(f = 2 \* precision \* recall / (precision + recall))

(specificity <- confusion\_matrix[1,1]/sum(confusion\_matrix[1,]))

library(ROCR)

train\_predictions <- predict(heart\_model, newdata = heart\_train,

type = "response")

perf <- performance(pred, measure = "prec", x.measure = "rec")

plot(perf)

thresholds <- data.frame(cutoffs = perf@alpha.values[[1]], recall =

perf@x.values[[1]], precision = perf@y.values[[1]])

subset(thresholds,(recall > 0.87) & (precision > 0.77))

Reference

* Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8).
* Lecture notes(Advanced Data Analytics Topics (ISM:663)) by professor Liping liu.