Connect, Message, Like,Follow&Share, 100% Free Counselling➔ThankYou

Logistic Regression in R and Python

Project to apply Logistic Regression

Problem Statement

HR analytics is revolutionizing the way human resources departments operate, leading to higher efficiency and better results overall. Human resources have been using analytics for years.

However, the collection, processing, and analysis of data have been largely manual, and given the nature of human resources dynamics and HR KPIs, the approach has been constraining HR. Therefore, it is surprising that HR departments woke up to the utility of machine learning so late in the game. Here is an opportunity to try predictive analytics in identifying the employees most likely to get promoted.

What is Logistic Regression?

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary/categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent

variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

Derivation of Logistic Regression Equation

Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (glm). In 1972, Nelder and Wedderburn proposed this model with an effort to provide a means of using linear regression to the problems which were not directly suited for application of linear regression. Infact, they proposed a class of different models (linear regression, ANOVA, Poisson Regression etc) which included logistic regression as a special case.

The fundamental equation of generalized linear model is:

$$g(E(y)) = \alpha + \beta x1 + \gamma x2$$

Here, g() is the link function, E(y) is the expectation of target variable and $\alpha + \beta x1 + \gamma x2$ is the linear predictor (α,β,γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor.

Important Points

1. GLM does not assume a linear relationship between dependent and independent variables. However, it assumes a linear relationship between link function and independent variables in logit model.
2. The dependent variable need not to be normally distributed.
3. It does not uses OLS (Ordinary Least Square) for parameter estimation. Instead, it uses maximum likelihood estimation (MLE).

4. Errors need to be independent but not normally distributed.

Let's understand it further using an example:

We are provided a sample of 1000 customers. We need to predict the probability whether a customer will buy (y) a particular magazine or not. As you can see, we've a categorical outcome variable, we'll use logistic regression.

To start with logistic regression, I'll first write the simple linear regression equation with dependent variable enclosed in a link function:

$$g(y) = \beta o + \beta(Age) \qquad \text{---- (a)}$$

Note: For ease of understanding, I've considered 'Age' as independent variable.

In logistic regression, we are only concerned about the probability of outcome dependent variable (success or failure). As described above, g() is the link function. This function is established using two things: Probability of Success(p) and Probability of Failure(1-p). p should meet following criteria:

1. It must always be positive (since p >= 0)
2. It must always be less than equals to 1 (since p <= 1)

Now, we'll simply satisfy these 2 conditions and get to the core of logistic regression. To establish link function, we'll denote g() with 'p' initially and eventually end up deriving this function.

Since probability must always be positive, we'll put the linear equation in exponential form. For any value of slope and dependent variable, exponent of this equation will never be negative.

$$p = \exp(\beta o + \beta(Age)) = e^{(\beta o + \beta(Age))} \quad \text{-------- (b)}$$

To make the probability less than 1, we must divide p by a number greater than p. This can simply be done by:

$$p = \exp(\beta o + \beta(Age)) / \exp(\beta o + \beta(Age)) + 1 = e^{(\beta o + \beta(Age))} / e^{(\beta o + \beta(Age))} + 1 \quad \text{----- (c)}$$

Using (a), (b) and (c), we can redefine the probability as:

$$p = e^{y}/ 1 + e^{y} \quad \text{--- (d)}$$

where p is the probability of success. *This (d) is the Logit Function*

If p is the probability of success, 1-p will be the probability of failure which can be written as:

$$q = 1 - p = 1 - (e^{y}/ 1 + e^{y}) \quad \text{--- (e)}$$

where q is the probability of failure

On dividing, (d) / (e), we get,

$$\frac{p}{1 - p} = e^{y}$$

After taking log on both side, we get,

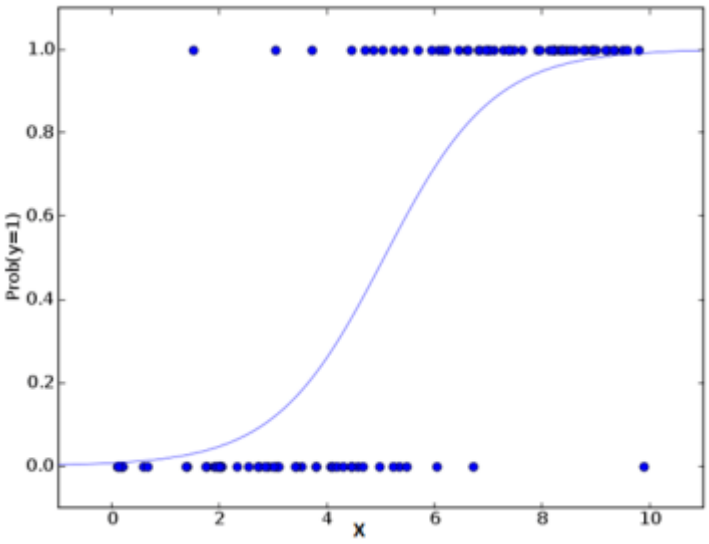
$$\log\left(\frac{p}{1 - p}\right) = y$$

log(p/1-p) is the link function. Logarithmic transformation on the outcome variable allows us to model a non-linear association in a linear way.

After substituting value of y, we'll get:

$$\log\left(\frac{p}{1-p}\right) = \beta_o + \beta(\text{Age})$$

This is the equation used in Logistic Regression. Here (p/1-p) is the odd ratio. Whenever the log of odd ratio is found to be positive, the probability of success is always more than 50%. A typical logistic model plot is shown below. You can see probability never goes below 0 and above 1.



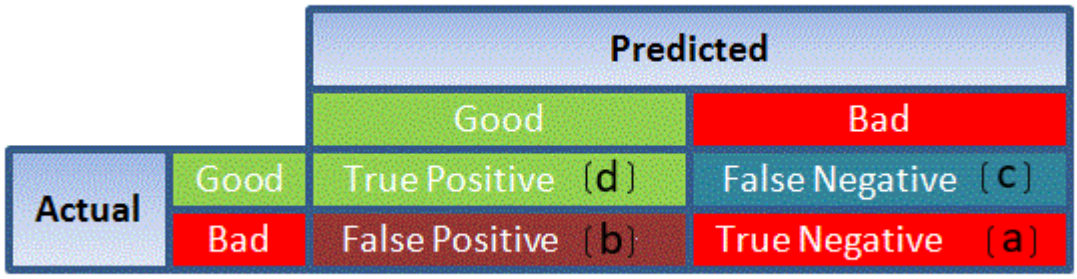
Performance of Logistic Regression Model

To evaluate the performance of a logistic regression model, we must consider few metrics. Irrespective of tool (SAS, R, Python) you would work on, always look for:

1. **AIC (Akaike Information Criteria)** – The analogous metric of adjusted R^2 in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with minimum AIC value.

2. **Null Deviance and Residual Deviance** – Null Deviance indicates the response predicted by a model with nothing but an intercept. Lower the value, better the model. Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.

3. **Confusion Matrix:** It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid overfitting. This is how it looks like:



Source: (plug – n – score)

You can calculate the **accuracy** of your model with:

$$\frac{\text{True Positive + True Negatives}}{\text{True Positive + True Negatives + False Positives + False Negatives}}$$

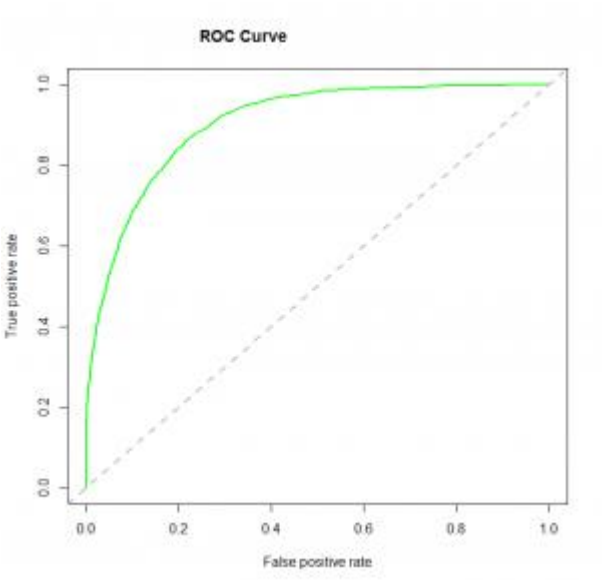
From confusion matrix, Specificity and Sensitivity can be derived as illustrated below:

$$\left.\begin{aligned}\text{True Negative Rate (TNR), specificity} &= \frac{A}{A+B}\\ \text{False Positve Rate (FPR), } 1-\text{specificity} &= \frac{B}{A+B}\end{aligned}\right\}\text{sum to 1}$$

$$\left.\begin{aligned}\text{True Positive Rate (TPR), sensitivity} &= \frac{D}{C+D}\\ \text{False Negative Rate (FNR)} &= \frac{C}{C+D}\end{aligned}\right\}\text{sum to 1}$$

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

4. **ROC Curve:** Receiver Operating Characteristic(ROC) summarizes the model's performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity). For plotting ROC, it is advisable to assume $p > 0.5$ since we are more concerned about success rate. ROC summarizes the predictive power for all possible values of $p > 0.5$. The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. This curve will touch the top left corner of the graph.



Note: For model performance, you can also consider likelihood function. It is called so, because it selects the coefficient values which maximizes the likelihood of explaining the observed data. It indicates goodness of fit as its value approaches one, and a poor fit of the data as its value approaches zero.

Logistic Regression Model in R and Python

The R code is provided below but if you're a Python user, here's an awesome code window to build your logistic regression model. No need to open Jupyter – you can do it all here:

Without going deep into feature engineering, here's the script of simple logistic regression model:

```
setwd('C:/Users/manish/Desktop/dressdata')
#load data
train <- read.csv('Train_Old.csv')

#create training and validation data from given data
```

```
install.packages('caTools')
library(caTools)

set.seed(88)
split <- sample.split(train$Recommended, SplitRatio = 0.75)

#get training and test data
dresstrain <- subset(train, split == TRUE)
dresstest <- subset(train, split == FALSE)

#logistic regression model
model <- glm (Recommended ~ .-ID, data = dresstrain, family = binomial)
summary(model)
predict <- predict(model, type = 'response')
#confusion matrix
table(dresstrain$Recommended, predict > 0.5)
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, dresstrain$Recommended)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
#plot glm
library(ggplot2)
ggplot(dresstrain, aes(x=Rating, y=Recommended)) + geom_point() +
stat_smooth(method="glm", family="binomial", se=FALSE)
```

This data require lots of cleaning and feature engineering.