Chapter – 02 Logistic Regression

Briefly we know M/L is an age old area of study and now with the availability of data and robust computing, M/L is feasible and corporations are able to harness value hidden in the data.

Our focus is classification, and in general classification algorithms try to construct a model to predict the class given a feature vector.

> tissue sample – benign OR malignant
>
> cc transaction – legitimate OR fraudulent
>
> credit application – approve OR deny
>
> candidate – hire OR reject
>
> student – admit OR reject
>
> email – spam OR not spam (ham)

In each example above there are two classes and are called binary classification. In contrast, imagine a scenario where the scanner must determine if it is an apple, peach, nectarines, pears, oranges or bananas. Or detect if it is a tiger, jaguar, lion, leopard, or a panther. Here classifier must determine a class out of many (not two) and such a problem is called multi-class classification. While a multi-class classifier can be deployed to solve a binary classification problem, binary classification algorithms cannot be directly employed to solve multi-class classification problem.

Therefore, we must study the given data to determine the class variable and whether it is multiclass or binary.

Process of Classification

The Lifecycle of a classification exercise can be divided into two top level phases 1) first a learning phase, aka training phase, where the algorithm is trained over labeled dataset. resulting in a model; The labeled dataset available for training is called training set 2) classification phase where the model is deployed over never seen before data to generate the missing labels for the new observations. During model engineering (pre-deployment) this phase, where a trained model is asked to label a never seen before observation, is called testing/validation phase. Post deployment,this phase is called scoring in the industry. Note that the model generalizes what it has learned during the induction phase to classify never seen before data. So all classification is inductive learning acquiring the ability to generalize.

Algorithm selection

While occam's razors guides us to select the simpler solution and the NFL theorem suggests no particular algorithm enjoys superior performance over the entire domain of problems, we only have heuristics toward algorithms. Recall, in Linear Regression, one relies on computed metrics such as R-square or RMSE to assess the quality of the linear regression model. Classifier output, a class label, is qualitative and not quantitative, and therefore RMSE or R-Square are not applicable. We need objective criteria to compare one classifier with another and to that end, let us define a few, appropriate for classification problems.

Consider a classification case, pregnancy test. The patient may be pregnant (positive case) and our model can correctly classify as pregnant (true positive, TP) or misclassify as not pregnant (FALSE negative FN). On the other hand, our patient may NOT be pregnant (negative case) and again our classifier can correctly classify as not pregnant (true negative, TN) or misclassify as pregnant (FALSE POSiTiVE, FP). Note that TP+FN+TN+FP must equal number of observations.

Therefore,

total positive P=TP+FN,

total negative N=TN+FP,

Proportion of correct classification, accuracy = (TP+TN)/(P+N)

Misclassification rate = (FP+FN)/(P+N)

Proportion of true positives, sensitivity = TP/P

Proportion of true positives, sensitivity = TN/N

Precision=TP/(TP+FP) If the algo says positive , how probable that actually is positive in reality

Recall = TP/(TP+FN)

Now we have an objective method to compare one algorithm with another. These are definitional and there are others – please refer to the slides.

Capabilities we need:

R and python are the tools of choice for data science. R and python are both equivalent but there appears to be two groups of uses. R is preferred by statistically bent and python if enterprise driven. We use R and you can download R from cran https://cran.r-project.org/. Then, once you gain proficiency, you can start with RStudio, an efficient IDE for R.

Generic capabilities

We need several capabilities before we can run the algorithms, regardless of the algorithms.

to load data.

to determine the class attribute and the number of classes

to perform preliminary EDA –for exploratory data analysis.

Prepare (transform) data for algorithm

Loading data

For the remainder of our journey we will use either heart.csv or ecoli.csv.

In R, we can load that data into a dataframe object with the name data.

```
data<-read.csv(file='heart.csv',head=F,sep=',',stringsAsFactors=F)

typeof(data)
```

```
class(data)

nrow(data)

ncol(data)

dim(data)

names(data)
```

Finding the class variable and determining the number of classes

The owner (business unit) must specify what they want to predict. Let us assume that we are asked to predict 'target'.

```
names(data) # lists all the col names -- which (names(heart)=='target')

NV<-names(data) #NV is a list of names, let us assume there is a col named target

which(NV=='target') # position of the class target  label column

# find spectral count

table(data$target)

names(table(data$target))

as.numeric(table(data$target))

length(names(table(data$target))) ## if this is 2 then binary classification

print(ifelse(length(names(table(data$target)))==2,"Binary Classification",

 "MultiClass Classification"))
```

Categorical vs numerical value

Note that a feature may have numerical values but it may not be numerical. For example, Male gender may be coded as 1 and female as 0. Here 1 and 0 are not numeric.

Columns are categorical if the values are not numeric , and columns are categorical even if the columns are numerical, but there are a finite set of values, as in the case of {gender :0 OR 1 }.

Let us review the relationships between predictors and class variable

We want to eliminate highly correlated predictors and constant predictors.

Let us plot the predictors one vs the other if any of them are redundant

```
pairs(heart)
```

Do we need two highly correlated predictor variables? We can drop one of them, without loss of learning or predictive ability.

Eliminate constant predictors

```
COLS<-unlist(lapply(1:13,FUN=function(x) {
```

```
TBL<-table(heart[[x]])

ifelse(length(names(TBL))<2,-1*x,x)}))

heartX<-heart[,COLS>0]
```

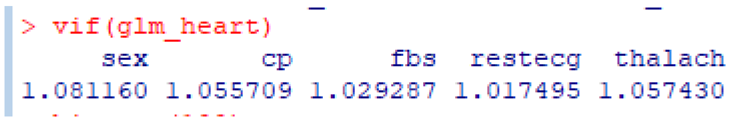
Relationship between Predictors and the class variable

Let us plot each predictor vs the target

```
cp<-par(mfrow=c(3,5)) #mulitple charts in one plot,

lapply(1:13,FUN=function(x)plot(heart[,x],heart[,14]))
```

Note that if two variables are highly correlated we do not need to include both in the model. This is known as multi-collinearity. There are standard techniques to get rid of multi-collinearity. R comes with many standard packages to determine importance of variables. We show here how to use vif function.

```
heart_scaled<-scale(heart[,which(names(heart)!="target")])

heart.df=as.data.frame(heart_scaled[,c(2,3,6,7,8)])

heart.df$target<-heart$target

head(heart.df)

glm_heart<-glm(target~.,data=heart.df,family='binomial')

if (!require(car)) install.packages('car')

library(car)

vif(heart_glm)

vif(glm_heart)
```



```
> vif(glm_heart)
     sex       cp      fbs  restecg  thalach
1.081160 1.055709 1.029287 1.017495 1.057430
```

VIF above 5 is trouble and since I have selected columns avoiding multicollinearity we do not see multi-collinearity. Please see https://www.statology.org/variance-inflation-factor-r/ to effortlessly visualize VIF. For scaling (standardization and normalization) read https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Datasets are multivariate and mostly rectangular

The number of observations (often denoted either as N or n) is never the same as the number of predictors (p). And p is almost always more than 1, so our dataset is multivariate and rectangular.

High dimensions (p>>n) results in significant challenge but it is out of scope. Interested students may search for Curse of Dimensionality and Dimensionality Reduction Techniques.

Since the domain of our variables are both numerical and categorical, and because many algorithms employ linear algebraic techniques (involving matrix operations) strategies, the need to encode the categorical columns into a numeric form, may become inevitable. We will defer this task until it becomes necessary.

Our heart.csv dataset is multivariate and different variables have different range,mean,standard deviation as shown by the following R commands, using base functions sd for standard deviation(σ), mean (μ), and range.

```
round(apply(heart,2,sd),4) # standard deviation by column

apply(heart,2,range) # min/max for each column

round(apply(heart,2,mean),4) # mean for each column
```

Variables can be scaled (converted to μ=0 and σ=1) as follows x_scaled=(x-mean(x))/sd(x).

Entire datasets can be scaled with scale function

```
heart_scaled<-scale(heart[,which(names(heart)!="target")])
```

It is not necessary to scale categorical features and therefore no need to scale the target, in classification problems.

Note, we have to be mindful if we transform or encode. When new data is to be categorized, will be in the original space and the new data will have to be transformed (scaled or normalized) appropriately to predict. The transformation constants have to be stored (such as μ and σ).

Splitting data

We must select the model that generalizes best. That is, the model that learns from given data and makes best inferences from never seen before data. And we don't want to wait for new data until model is deployed to estimate how the model performs over never seen before data. So to simulate that we choose not to train on the entire dataset, but "hold out" some portion of the given data, say 70/30. We shall split the data into 70% training data and 30 % test data. We train the algorithm and develop a model on 70% training data and confirm model is performant using the test data.

In R, we can split the dataset effortlessly as follows:

```
# Prior to splitting the data, perform all scaling and encoding so test and train have the same scaling factors.

set.seed(43)

tridx<-sample(1:nrow(dataset),0.7*nrow(dataset),replace=F)

trdf<-dataset[tridx,]

tstdf<-dataset[-tridx,]

table(dataset$target) # run table to ensure class distributions are NOT overly impacted.

table(trdf$target) # need not be identical but somewhat similar
```

table(tstdf$target)

Reproducibility and Repeatability

Statistical inferences must be reproducible and repeatable. Care must be taken so random number series do not contribute to random results. So one should always ***set the seed*** so sample and other statistical computations generate the same output.

Overfitting and underfitting

Error on the test data is our generalization error. If the model performs well over the training data (perform here means model is able to classify correctly) but decreases over the test data – then it is attributed to over-fitting. The model has captured the noise along with the signal and it is not generalizing incorrectly. If on the other hand, the model is too simple and is unable to learn the relationship between the feature vectors and the labels even when the labels are given, then the model is under-fitting. For example if our model is y=mode(y) or some constant or a linear model to predict a quadratic function, under-fitting will result.

Bias and Variance

Total Error can be shown to be Error=var+bias^2+err from first principles of Expected Values of variance, bias and an irreducible component err. Detailed derivation of this result is presented in many ML text books.

Modeling

Let us start with linear regression where the dependent variable (DV) the target is continuous variable – real. In the heart dataset, however, the target is either 0 or 1 – both real values but target has a restricted domain. Thus a standard linear regression model will not transform given input variables to such a restricted domain (an algebraic function to map real to dichotomous discrete domain does not exist). We need a transformation that can convert any real to value to 0 to 1. The sigmoid function transforms any real number to a number between 0 and 1. Sigmoid function, in R, is,

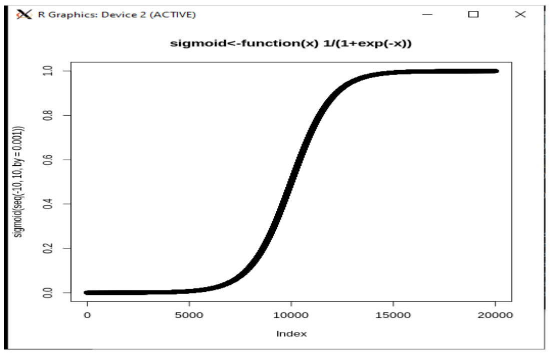
sigmoid<-function(z) 1/(1+exp(-z)).

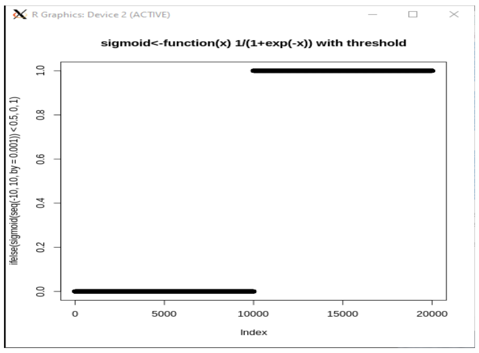
Let us plot the sigmoid function and review the transformation, for the range -10 to 10 incrementing by 0.001:

plot(sigmoid(seq(-10,10,by =0.001)),main="sigmoid<-function(z) 1/(1+exp(-z))").

We note that sigmoid returns all values in the range 0 and 1, not just 0 and 1.

Y = sigmoid(z) and here $Z= b_0+b_1x_1+b_2x_2+....b_nx_n$





Now we have a function that does what we want and let us find a way to integrate the data.

Let $z = b_0+b_1*x_1+b_2*x_2....+b_nx_n$

Using sigmoid(z) we transform the feature vector to a value 0 to 1. Transformation is complete. But we need 0 or 1 representing the two classes. We can recast the sigmoid(z) as the odds ratio and derive a linear model. Derivation of this odds ratio is simple algebra, as shown below. Note that,

1/(1+exp(-z))=exp(z)/(exp(z)+1).

$$P(Y=1|X) = \frac{e^{(\beta_o+\beta_1 x)}}{e^{(\beta_o+\beta_1 x)}+1}$$

https://www.hackerearth.com/blog/wp-content/uploads/2017/01/equateimage.png

NY EMAIL ISL Statistics How to Stop Skype Fr Attributes: Defining C Fundry & Yesco - Firm

$$\Longrightarrow \quad p(X) = \frac{e^{(\beta_o+\beta_1 x)}}{e^{(\beta_o+\beta_1 x)}+1}$$

$$\Longrightarrow \quad p(e^{(\beta_o+\beta_1 x)}+1) = e^{(\beta_o+\beta_1 x)}$$

$$\Longrightarrow \quad p.e^{(\beta_o+\beta_1 x)}+p = e^{(\beta_o+\beta_1 x)}$$

$$\Longrightarrow \quad p = e^{(\beta_o+\beta_1 x)} - p.e^{(\beta_o+\beta_1 x)}$$

$$\Longrightarrow \quad p = e^{(\beta_o+\beta_1 x)}(1-p)$$

$$\Longrightarrow \quad \frac{p}{1-p} = e^{(\beta_o+\beta_1 x)}$$

$$\Longrightarrow \quad \ln(\tfrac{p}{1-p}) = \beta_0 + \beta_1 x$$

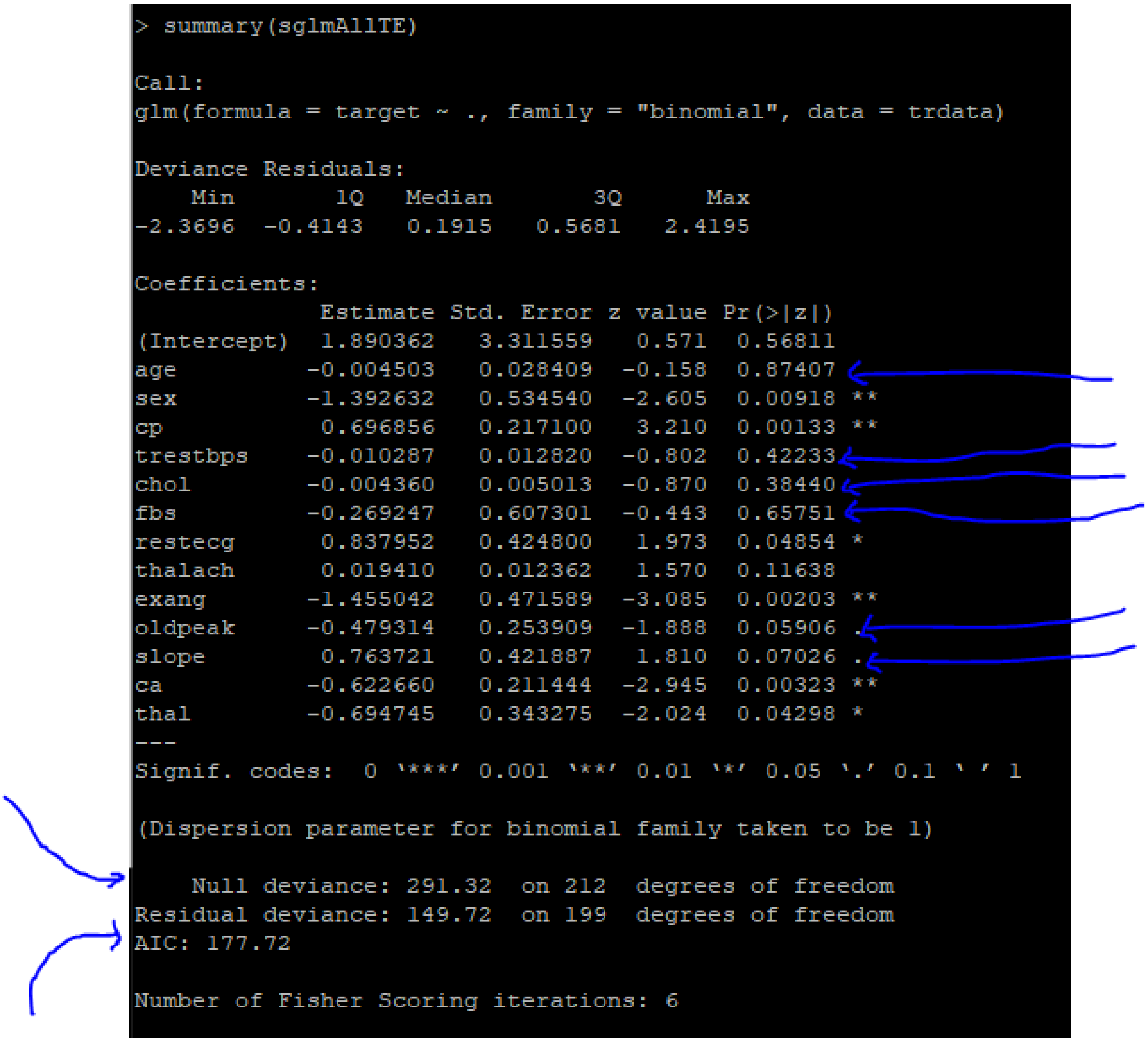
Our challenge is to find bn,b2,b1,b0 the betas. Now we have a familiar representation of linear regression. Hence this classifier is called logistic regression. Called parameters,the β can be estimated using standard techniques such as Gradient Descent or MLE. We will study both GD and MLE in detail later in the semester.

We run a regression, estimate the betas and compute the log odds from which probability p and (1-p) can be computed. The class with the higher probability is assigned to the new instance.

For now we will learn to use one of the many implementations of logistic regression readily available with standard R installation,called *glm* generalized linear model. So let us run logistic regression on the heart dataset as shown below:

R Script is included in the R file.

Let us run summary to review the validity of our model



```
> summary(sglmAllTE)

Call:
glm(formula = target ~ ., family = "binomial", data = trdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3696  -0.4143   0.1915   0.5681   2.4195

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.890362   3.311559   0.571  0.56811
age         -0.004503   0.028409  -0.158  0.87407
sex         -1.392632   0.534540  -2.605  0.00918 **
cp           0.696856   0.217100   3.210  0.00133 **
trestbps    -0.010287   0.012820  -0.802  0.42233
chol        -0.004360   0.005013  -0.870  0.38440
fbs         -0.269247   0.607301  -0.443  0.65751
restecg      0.837952   0.424800   1.973  0.04854 *
thalach      0.019410   0.012362   1.570  0.11638
exang       -1.455042   0.471589  -3.085  0.00203 **
oldpeak     -0.479314   0.253909  -1.888  0.05906 .
slope        0.763721   0.421887   1.810  0.07026 .
ca          -0.622660   0.211444  -2.945  0.00323 **
thal        -0.694745   0.343275  -2.024  0.04298 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 291.32  on 212  degrees of freedom
Residual deviance: 149.72  on 199  degrees of freedom
AIC: 177.72

Number of Fisher Scoring iterations: 6
```

Let us now compute the metrics during training and testing to detect any sign of over/under fitting.

It is left as an exercise for you to try out glm with (sex,cp,restecg,thalach,ca,thal)

R comes loaded with utilities so let us indulge, shall we…

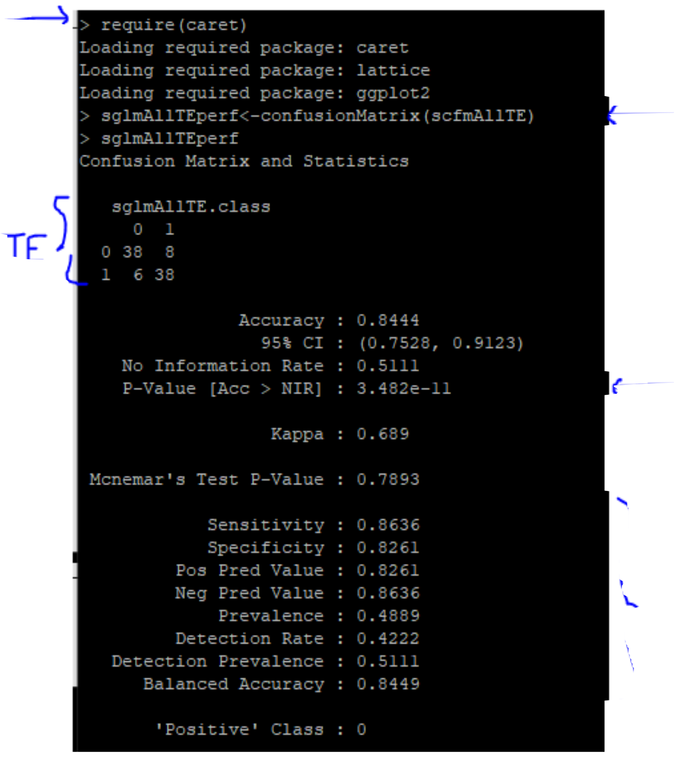
Let us load caret package and compute all the metrics (accuracy, sensitivity, specificity, recall and precision) for the test and training phase and evaluate it for under/over-fitting.

In comparison to training, accuracy has dropped from 0.8548 to 0.8444,as we would expect. Yet, there is no need to be alarmed about overfitting. Performance during test phase (never seen before samples) cannot exceed training phase performance (all known samples).

Logistic Regression is a bell weather classifier. One should always start with LR but definitely not stop with it. Use that as a benchmark. Run other varied classifiers and evaluate the performance at a deeper level.

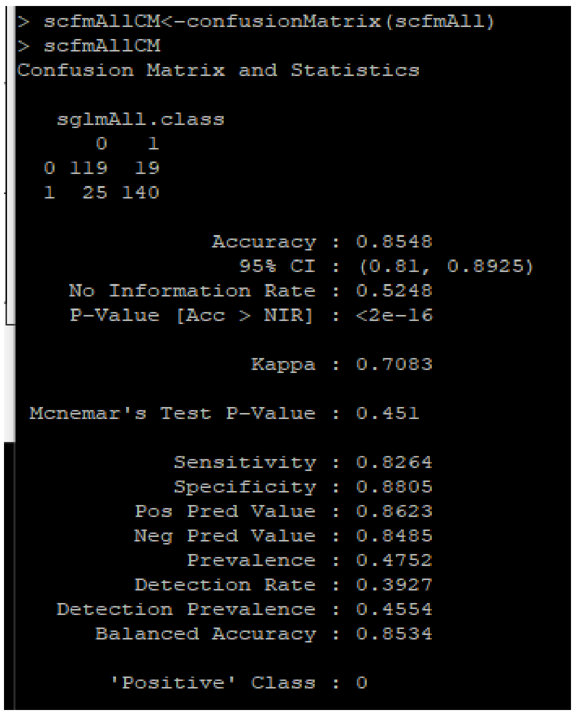
Next we will develop the intuition for NaiveBayes where we will assume that the predictors are conditionally independent. We will use chain-rule of probabilities. Naïve Bayes is a classic probabilistic classifier.

Metrics from test set.



```
> require(caret)
Loading required package: caret
Loading required package: lattice
Loading required package: ggplot2
> sglmAllTEperf<-confusionMatrix(scfmAllTE)
> sglmAllTEperf
Confusion Matrix and Statistics

   sglmAllTE.class
     0  1
  0 38  8
  1  6 38

               Accuracy : 0.8444
                 95% CI : (0.7528, 0.9123)
    No Information Rate : 0.5111
    P-Value [Acc > NIR] : 3.482e-11

                  Kappa : 0.689

 Mcnemar's Test P-Value : 0.7893

            Sensitivity : 0.8636
            Specificity : 0.8261
         Pos Pred Value : 0.8261
         Neg Pred Value : 0.8636
             Prevalence : 0.4889
         Detection Rate : 0.4222
   Detection Prevalence : 0.5111
      Balanced Accuracy : 0.8449

       'Positive' Class : 0
```

Metrics from training.



```
> scfmAllCM<-confusionMatrix(scfmAll)
> scfmAllCM
Confusion Matrix and Statistics

   sglmAll.class
      0   1
  0 119  19
  1  25 140

               Accuracy : 0.8548
                 95% CI : (0.81, 0.8925)
    No Information Rate : 0.5248
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7083

 Mcnemar's Test P-Value : 0.451

            Sensitivity : 0.8264
            Specificity : 0.8805
         Pos Pred Value : 0.8623
         Neg Pred Value : 0.8485
             Prevalence : 0.4752
         Detection Rate : 0.3927
   Detection Prevalence : 0.4554
      Balanced Accuracy : 0.8534

       'Positive' Class : 0
```