Logistic Regression

-Applied Analytics-

APM Chapter 12.2, ISLR Chapter 4.3

Lecturer: Darren Homrighausen, PhD

Introductory example

An Introductory Example

Suppose we work for a credit card company and we wish to identify people that are likely to default on their credit card debt

We have features (for 10,000 people):

- Student status
- Income
- Balance

Along with their default status:

$$Y = \begin{cases} 1 \text{ if person defaults} \\ 0 \text{ if person doesn't default} \end{cases}$$

Let's look at some plots.

An Introductory Example

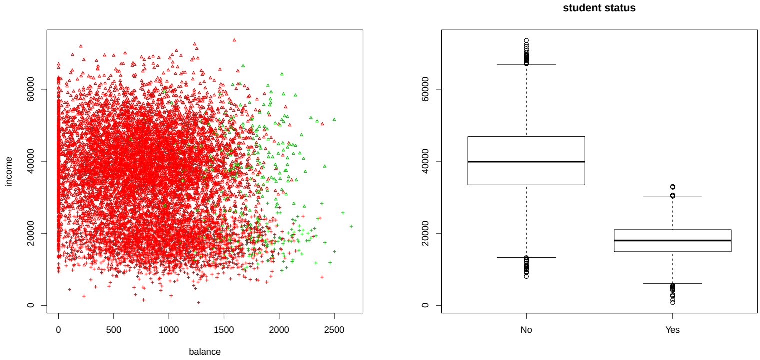
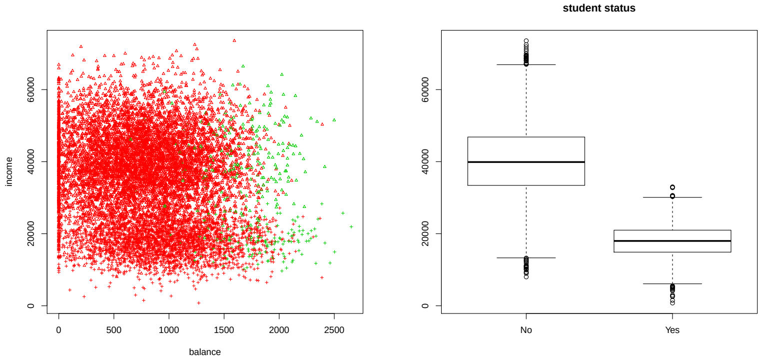


Figure: The red are people without defaults, green are defaults. The '+' are students, the 'Δ' are not students.

An Introductory Example



Some comments:

- Income doesn't seem to be related to defaults
- Student status is also unrelated to defaults, but highly related to income
- Balance seems to strongly predict default status.

An Introductory Example: Why not Use Regression?

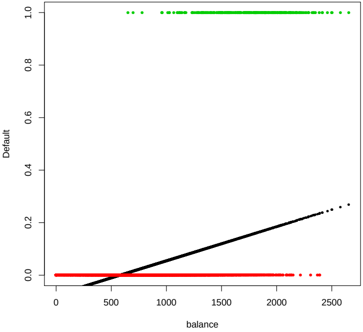
Suppose for a moment we only consider balance. Then, we can run a simple linear regression of default status on balance

```
Y = rep(0,n)
Y[default == 'Yes'] = 1
out.lm = lm(Y~balance)
summary(out.lm)
```

R will happily do this.

An Introductory Example: Why not Use Regression?

Let's plot our data with the fitted values (black) and the training supervisor (red/green)



Not so great..

Logistic regression

Logistic regression

Logistic regression is a 'generalized linear model' (GLM)

The generalized means that the nature of the supervisor isn't naturally modeled as normal as it is in multiple linear regression:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \epsilon$$

(ϵ is modeled as a zero mean, symmetric $\rightarrow$ the residuals should look similar)

If we model probabilities, then we are doing Logistic regression

Logistic regression

Suppose for now that $C = 2$, $p_1 = p$, and $p_2 = 1 - p$

(The number of classes, and the probability that Y equals C_1 and C_2, respectively)

Then logistic regression models the probability as

$$p = \frac{\exp\{\beta_0 + \sum_{j=1}^{p} \beta_j x_j\}}{1 + \exp\{\beta_0 + \sum_{j=1}^{p} \beta_j x_j\}}.$$

This gets converted to something that looks like multiple regression via

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

(This is the logit function)

Interpreting the parameters

We can interpret the parameters in either linear model:

- Multiple linear regression.
  $f(X) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j$
  "A one unit change in x_j is associated with a β_j change in the mean of Y, holding all other features constant"
- Logistic regression. $\log\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$
  "A one unit change in x_j is associated with a β_j change in the log odds that $Y = C_1$, holding all other features constant"
  or
  "A one unit change in x_j is associated with a multiplicative $\exp\{\beta_j\}$ change in the odds that $Y = C_1$, holding all other features constant"

In either case, a $\beta_j > 0$ denotes a positive relationship and $\beta_j < 0$ a negative one

Estimation for logistic regression

With multiple regression, there is a closed form solution:

$$\hat{\beta} = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top Y$$

This was due to estimation via least squares

(This is also the maximum likelihood estimator (MLE) under ϵ being normally distributed)

To find the estimator $\hat{\beta}$ for logistic regression, we have to use an iterative maximization technique

Let's begin to look at the output:

Estimating logistic regression

```
out.glm  = glm(default~balance,family='binomial')
> summary(out.glm)
Call:
glm(formula = default ~ balance, family = "binomial")
Deviance Residuals:
    Min       1Q    Median      3Q      Max
-2.2697  -0.1465  -0.0589  -0.0221   3.7589
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
...
Number of Fisher Scoring iterations: 8
```

Predictions: Probability estimates

Once we have the $\hat{\beta}$, we can estimate the probabilities via

$$\hat{p} = \frac{\exp\{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j\}}{1 + \exp\{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j\}}.$$

Example: Suppose we want to predict the probability that someone with a balance of $1,000 defaults

$$\hat{p}(1000) = \frac{\exp\{-10.65 + 0.0055 * 1000\}}{1 + \exp\{-10.65 + 0.0055 * 1000\}} = 0.0058$$

Maybe look at $2,000 instead:

$$\hat{p}(2000) = \frac{\exp\{-10.65 + 0.0055 * 2000\}}{1 + \exp\{-10.65 + 0.0055 * 2000\}} = 0.586$$

Predictions: Classifications

We need to turn the estimated probabilities:

$$\hat{p} = \frac{\exp\{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j\}}{1 + \exp\{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j\}}.$$

into classifications

Example: Using the threshold of $1/C = 0.5$,

$$\hat{p}(1000) = \frac{\exp\{-10.65 + 0.0055 * 1000\}}{1 + \exp\{-10.65 + 0.0055 * 1000\}} = 0.0058$$

Thus, a balance of \$1000 would be classified as no default

Maybe look at \$2000 instead:

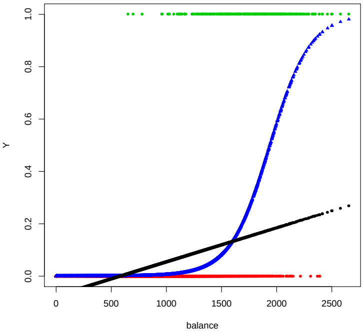
$$\hat{p}(2000) = \frac{\exp\{-10.65 + 0.0055 * 2000\}}{1 + \exp\{-10.65 + 0.0055 * 2000\}} = 0.586.$$

A balance of \$2000 would be classified as default

A comparison

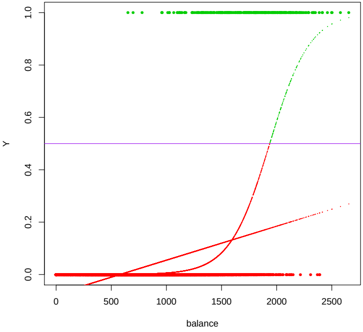
Let's plot our data with

- Simple Linear Regression (black)
- Logistic regression (blue)



A comparison

Results of using a cut-off of 0.5



Judging Model Performance

Using caret

```
trControl     = trainControl(method = 'cv', number = 10)
outLogistic   = train(x = Xtrain, y = Ytrain,
                  method = 'glm', trControl = trControl)
```

Let's get a training/test split as well, resulting in Xtrain, Xtest, Ytrain, and Ytest

(Use createDataPartition)

Getting predictions

```
YhatTestProb = predict(outLogistic, Xtest, type = 'prob')
> head(YhatTestProb)
           Yes        No
1 0.0005292166 0.9994708
2 0.0025568958 0.9974431
3 0.0023380867 0.9976619
4 0.0205685913 0.9794314
5 0.0001117464 0.9998883
6 0.0007992407 0.9992008
YhatTest = predict(outLogistic, Xtest, type = 'raw')
> head(YhatTest)
[1] No No No No No No
Levels: Yes No
```

Well calibrated probabilities

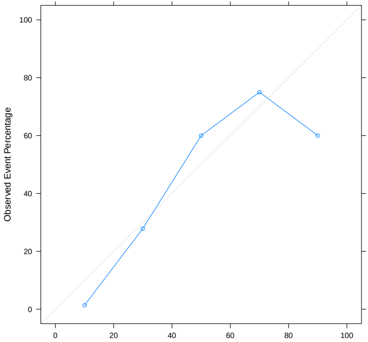
Question: What is a calibration plot? What are well-calibrated probabilities?

Well calibrated probabilities

Question: What is a calibration plot? What are well-calibrated probabilities?

```
calibProbs = calibration(Ytest~YhatTestProb$Yes, cuts = 5)

which(YhatTestProb$Yes > .8)
[1]  987 1004 1216 1735 2059
Ytest[which(YhatTestProb$Yes > .8)]
[1] Yes No  No  Yes Yes
```

Well calibrated probabilities




```
Ytest[which(YhatTestProb$Yes > .8)]
[1] Yes No  No  Yes Yes
```

These probability estimates are overall well calibrated

Rare 'Yes' events make the upper probability range unreliable

Confusion matrices

Using caret, we can report a lot of output

```
> confusionMatrix(data = YhatTest,
                  reference = Ytest)
Confusion Matrix and Statistics
          Reference
Prediction  Yes   No
       Yes   24   12
       No    59 2404

               Accuracy : 0.9716
                    ...
                  Kappa : 0.3911
                    ...
            Sensitivity : 0.289157
            Specificity : 0.995033
                    ...
       'Positive' Class : Yes
```

Adjusting the threshold

The caret package only allows us to use the $1/C$ threshold:

```
YhatTest = predict(outLogistic, Xtest, type = 'raw')
```

We can generate classifications out of the $\hat{p}$ with a different threshold:

```
YhatTestThresh = ifelse(YhatTestProb$Yes > .2,
                        'Yes', 'No')  %>%
  as.factor %>%
  relevel(ref = 'Yes')
```

The relevel command is needed to keep the levels consistent

```
> levels(Ytest)
[1] "Yes" "No"
> levels(YhatTestThresh)
[1] "Yes" "No"
```

Question: Do you expect sensitivity to go up or down?

Confusion matrices

For the threshold 0.2, we get:

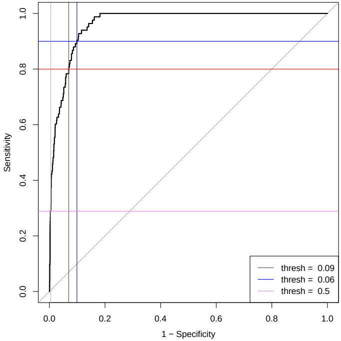
```
> confusionMatrix(data =  YhatTestThresh,
                reference = Ytest)
Confusion Matrix and Statistics
          Reference
Prediction  Yes   No
       Yes   50   60
       No    33 2356

               Accuracy : 0.9628
                    ...
                  Kappa : 0.4992
                    ...
            Sensitivity : 0.60241
            Specificity : 0.97517
                    ...
       'Positive' Class : Yes
```

ROC curve

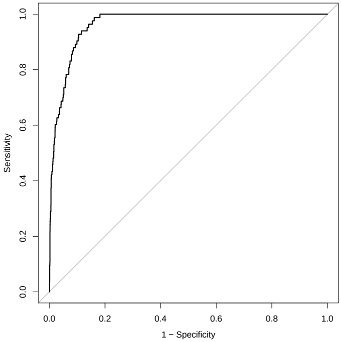
We can scan over all the available thresholds with the ROC curve

```
> levels(Ytest)
[1] "Yes" "No"
```

```
rocCurve = roc(relevel(Ytest,'No'), YhatTestProb$Yes)
```



AUC



The area under the curve (AUC) for this classifier is a one number summary of the ROC curve:

```
> rocCurve$auc
```

```
[1] 0.952
```

This would be considered a very good AUC score