Unit 3 Lecture 1: Logistic Regression

October 5, 2021

```
library(pROC)        # for ROC curves
library(tidyverse)
```

In today's R demo, we will apply logistic regression to the `Default` data from lecture:

```
default_data = ISLR2::Default %>% as_tibble()
default_data
```

```
## # A tibble: 10,000 x 4
##    default student balance income
##    <fct>   <fct>     <dbl>  <dbl>
##  1 No      No         730. 44362.
##  2 No      Yes        817. 12106.
##  3 No      No        1074. 31767.
##  4 No      No         529. 35704.
##  5 No      No         786. 38463.
##  6 No      Yes        920.  7492.
##  7 No      No         826. 24905.
##  8 No      Yes        809. 17600.
##  9 No      No        1161. 37469.
## 10 No      No           0  29275.
## # ... with 9,990 more rows
```

As an exploratory question, what is the default rate in this data?

The rest of the activity will be easier if we code `default` as 0-1:

```
default_data = default_data %>% mutate(default = as.numeric(default == "Yes"))
default_data
```

```
## # A tibble: 10,000 x 4
##    default student balance income
##      <dbl> <fct>     <dbl>  <dbl>
##  1       0 No         730. 44362.
##  2       0 Yes        817. 12106.
##  3       0 No        1074. 31767.
##  4       0 No         529. 35704.
##  5       0 No         786. 38463.
##  6       0 Yes        920.  7492.
##  7       0 No         826. 24905.
##  8       0 Yes        809. 17600.
##  9       0 No        1161. 37469.
## 10       0 No           0  29275.
## # ... with 9,990 more rows
```

Let's split the default data into training and test sets:

```
set.seed(471)
train_samples = sample(1:nrow(default_data), 0.8*nrow(default_data))
default_train = default_data %>% filter(row_number() %in% train_samples)
default_test = default_data %>% filter(!(row_number() %in% train_samples))
```

Running a logistic regression

The way to run a logistic regression is through the `glm` function:

```
glm_fit = glm(default ~ student + balance + income,
              family = "binomial",
              data = default_train)
summary(glm_fit)
```

```
##
## Call:
## glm(formula = default ~ student + balance + income, family = "binomial",
##     data = default_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1433  -0.1416  -0.0541  -0.0195   3.7396
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.106e+01  5.537e-01 -19.972  < 2e-16 ***
## studentYes  -7.051e-01  2.663e-01  -2.648  0.00811 **
## balance      5.824e-03  2.595e-04  22.444  < 2e-16 ***
## income       5.806e-06  9.112e-06   0.637  0.52399
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2394.2  on 7999  degrees of freedom
## Residual deviance: 1259.9  on 7996  degrees of freedom
## AIC: 1267.9
##
## Number of Fisher Scoring iterations: 8
```

Interpreting the estimates

- What is the coefficient estimate for `student`?
- Does this suggest that being a student increases or decreases the probability of default, other things being equal?
- According to this estimate, how does being a student impact the log-odds of default? How does it impact the odds of default?

Extracting elements of the fit

We can extract the coefficient estimates, standard errors, etc. just as we did with linear models:

```
coef(glm_fit)
```

```
##   (Intercept)    studentYes       balance        income 
## -1.105920e+01 -7.050610e-01  5.824362e-03  5.806162e-06 
```

Fitted probabilities and making predictions

We can extract the fitted probabilities of default for a test set using the `predict` function:

```r
fitted_probabilities = predict(glm_fit,
        newdata = default_test,
        type = "response")                # to get output on probability scale
head(fitted_probabilities)
```

```
##            1            2            3            4            5            6 
## 8.830647e-06 1.167429e-02 1.082938e-04 1.860563e-04 8.007356e-05 7.896278e-04 
```

We can now make predictions based on the fitted probabilities using the standard 0.5 threshold:

```r
predictions = as.numeric(fitted_probabilities > 0.5)
head(predictions)
```

```
## [1] 0 0 0 0 0 0
```

Evaluating the classifier

Let's calculate the misclassification rate of the above logistic regression classifier.

```r
# first add predictions to the tibble
default_test = default_test %>%
  mutate(predicted_default = predictions)
default_test
```

```
## # A tibble: 2,000 x 5
##    default student balance income predicted_default
##      <dbl> <fct>     <dbl>  <dbl>             <dbl>
##  1       0 Yes          0  21871.                 0
##  2       0 No        1113. 23810.                 0
##  3       0 No         286. 45042.                 0
##  4       0 Yes        528. 17637.                 0
##  5       0 No         229. 50500.                 0
##  6       0 No         642. 30466.                 0
##  7       0 No         773. 34353.                 0
##  8       0 Yes        221. 16873.                 0
##  9       0 No         409. 54207.                 0
## 10       0 No        1228. 37409.                 0
## # ... with 1,990 more rows
```

```r
# then calculate misclassification rate
default_test %>%
  summarise(mean(default != predicted_default))
```

```
## # A tibble: 1 x 1
##   `mean(default != predicted_default)`
##                                  <dbl>
## 1                                0.029
```

To get a fuller picture, let's calculate the confusion matrix:

```
default_test %>%
  select(default, predicted_default) %>%
  table()
```

```
##          predicted_default
## default     0    1
##         0 1931   11
##         1   47   11
```

- What are the false positive and false negative rates of this classifier?
- If the cost of a false negative is three times that of a false positive, what probability threshold should we use? What are the false positive and false negative rates for the resulting classifier?

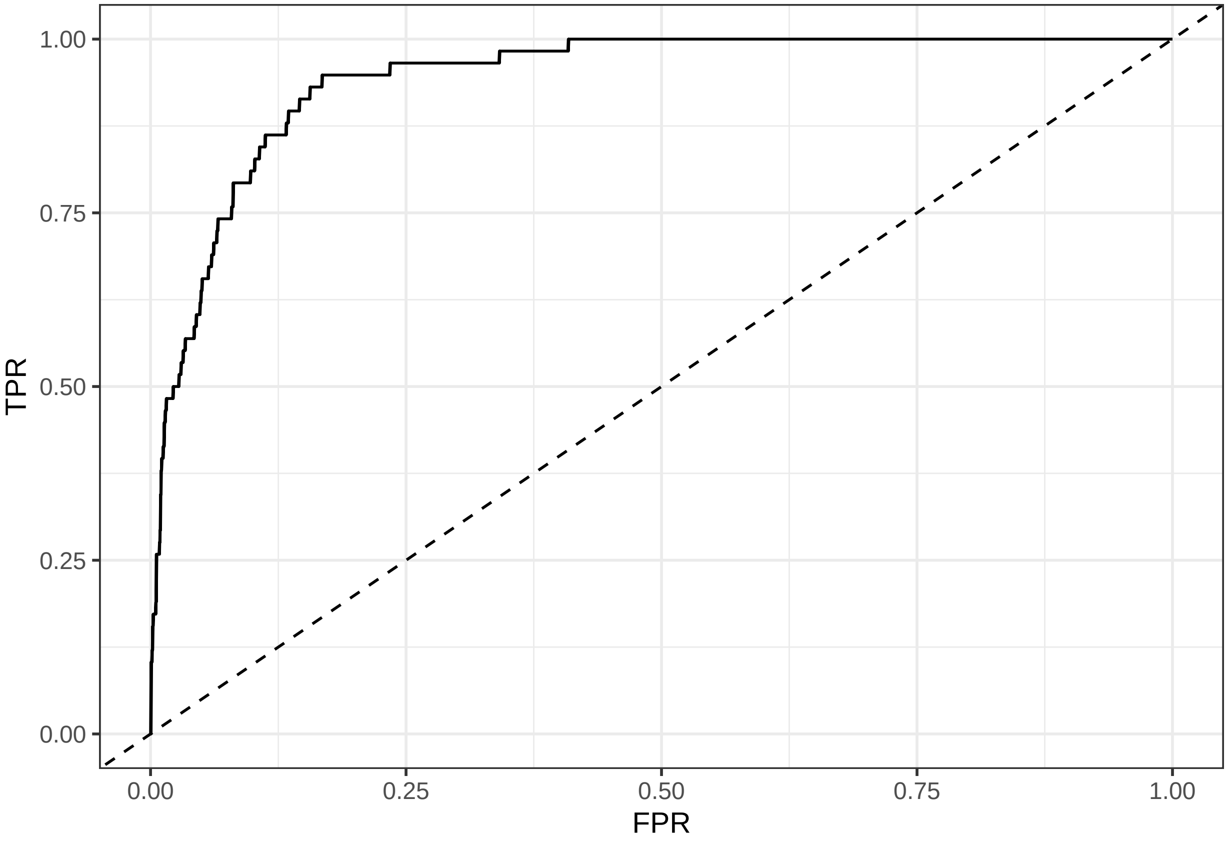
Next, let's plot the ROC curve for this classifier.

```
# ROC curve
roc_data = roc(default_test %>% pull(default),
               fitted_probabilities)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
tibble(FPR = 1-roc_data$specificities,
       TPR = roc_data$sensitivities) %>%
  ggplot(aes(x = FPR, y = TPR)) +
  geom_line() +
  geom_abline(slope = 1, linetype = "dashed") +
#  geom_point(x = fpr, y = 1-fnr, colour = "red") +
  theme_bw()
```



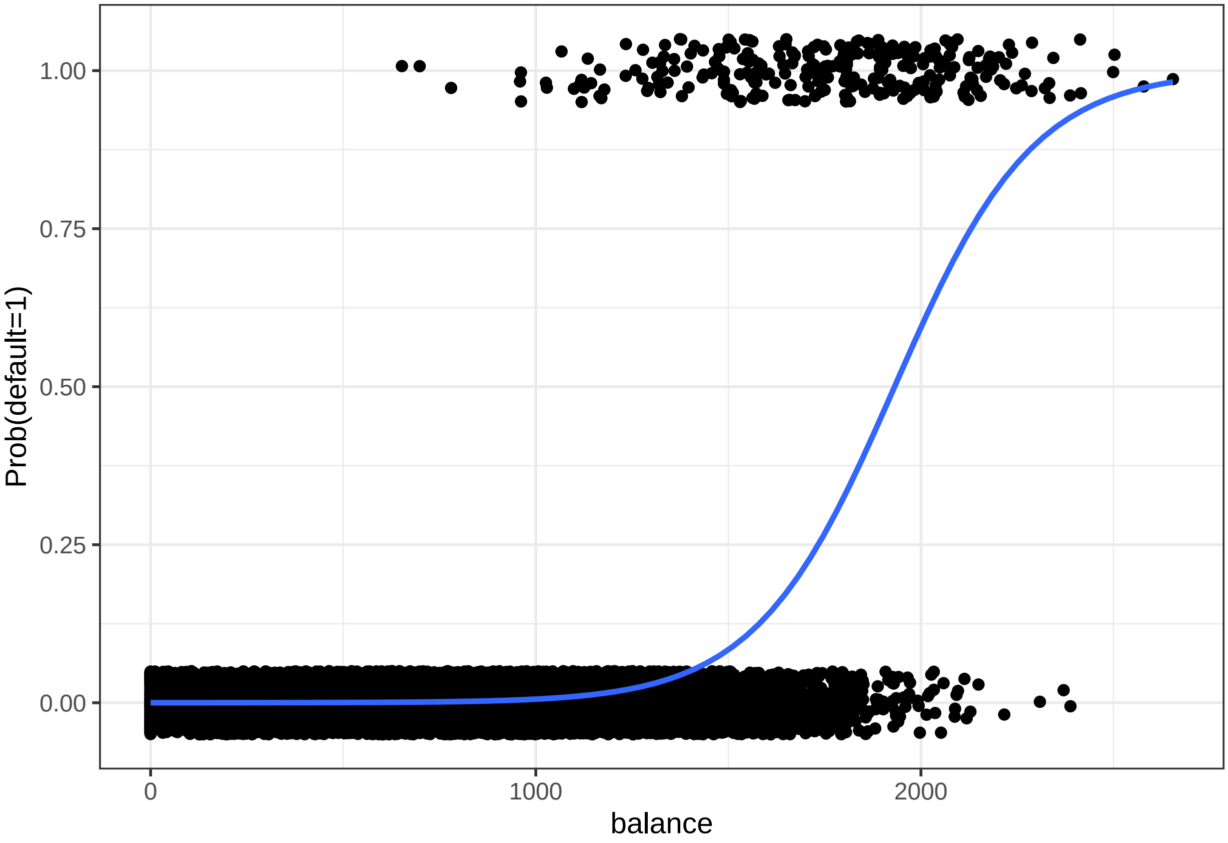
```
# print the AUC
roc_data$auc
```

```
## Area under the curve: 0.9438
```

Plotting a univariate logistic regression fit

Univariate logistic regression fits can be plotted using `geom_smooth`:

```
default_train %>%
  ggplot(aes(x = balance, y = default))+
  geom_jitter(height = .05) +
  geom_smooth(method = "glm",
              formula = "y~x",
              method.args = list(family = "binomial"),
              se = FALSE) +
  ylab("Prob(default=1)") +
  theme_bw()
```



Roughly at what value of balance do we switch from predicting no default to predicting default?