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
```

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
```

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

```
default_data %>%
  summarise(mean(default))
```

```
## # A tibble: 1 x 1
##   `mean(default)`
##             <dbl>
## 1          0.0333
```

The default rate is about 3%.

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)
coef(glm_fit)
```

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

- 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?

The coefficient estimate is about -0.7. This suggests that being a student decreases the probability of default. Being a student decreases the log-odds of default by 0.7, so it multiplies the odds of default by exp(-0.7), which is about 0.5. In other words, being a student halves the odds of default.

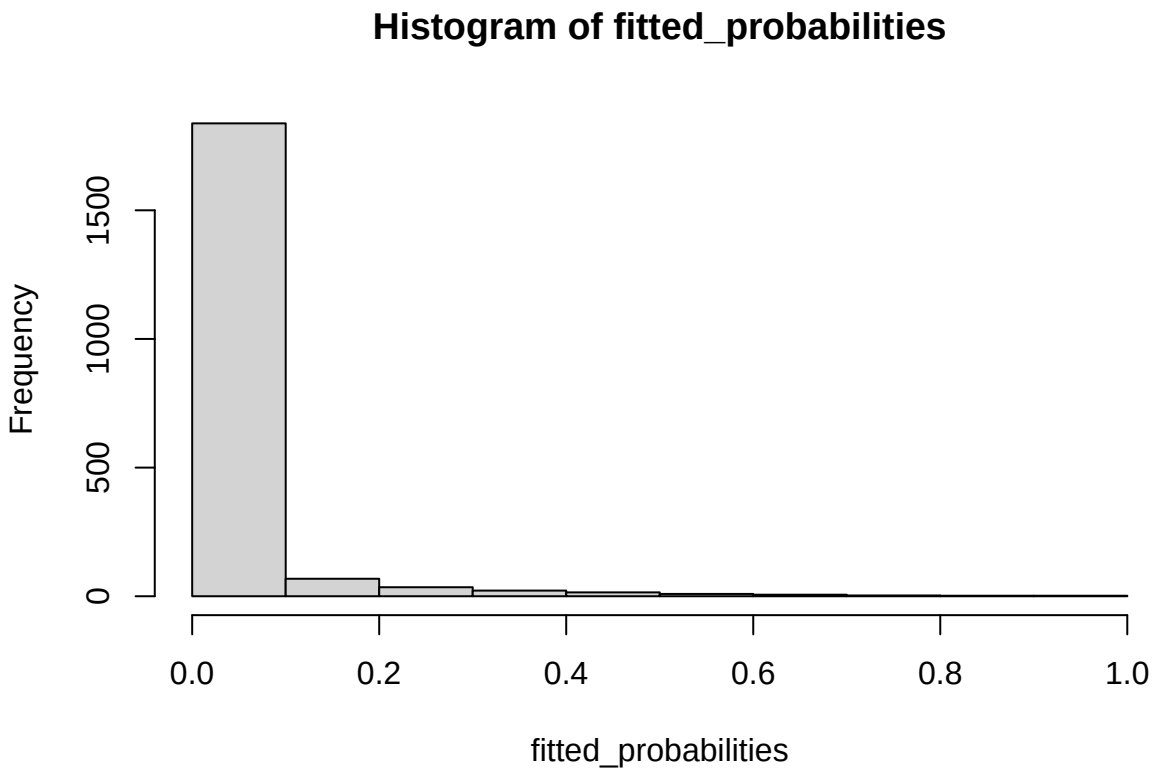
Fitted probabilities and making predictions

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

```
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
```

```
hist(fitted_probabilities)
```



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

```
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.

```
# 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
```

```
# 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?

The false positive rate is 11/(11+1931) = 0.006 and the false negative rate is 47/(47+11) = 0.81.

- 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?

```
thresh = 1/(1+3)
predictions = as.numeric(fitted_probabilities > thresh)
default_test %>%
  mutate(predicted_default = predictions) %>%
  select(default, predicted_default) %>%
  table()
```

```
##         predicted_default
## default    0    1
##       0 1899   43
##       1   30   28
```

We should use a threshold of 1/(1+3) = 1/4. The resulting false positive rate is about 2% and the resulting false negative rate is about 52%.

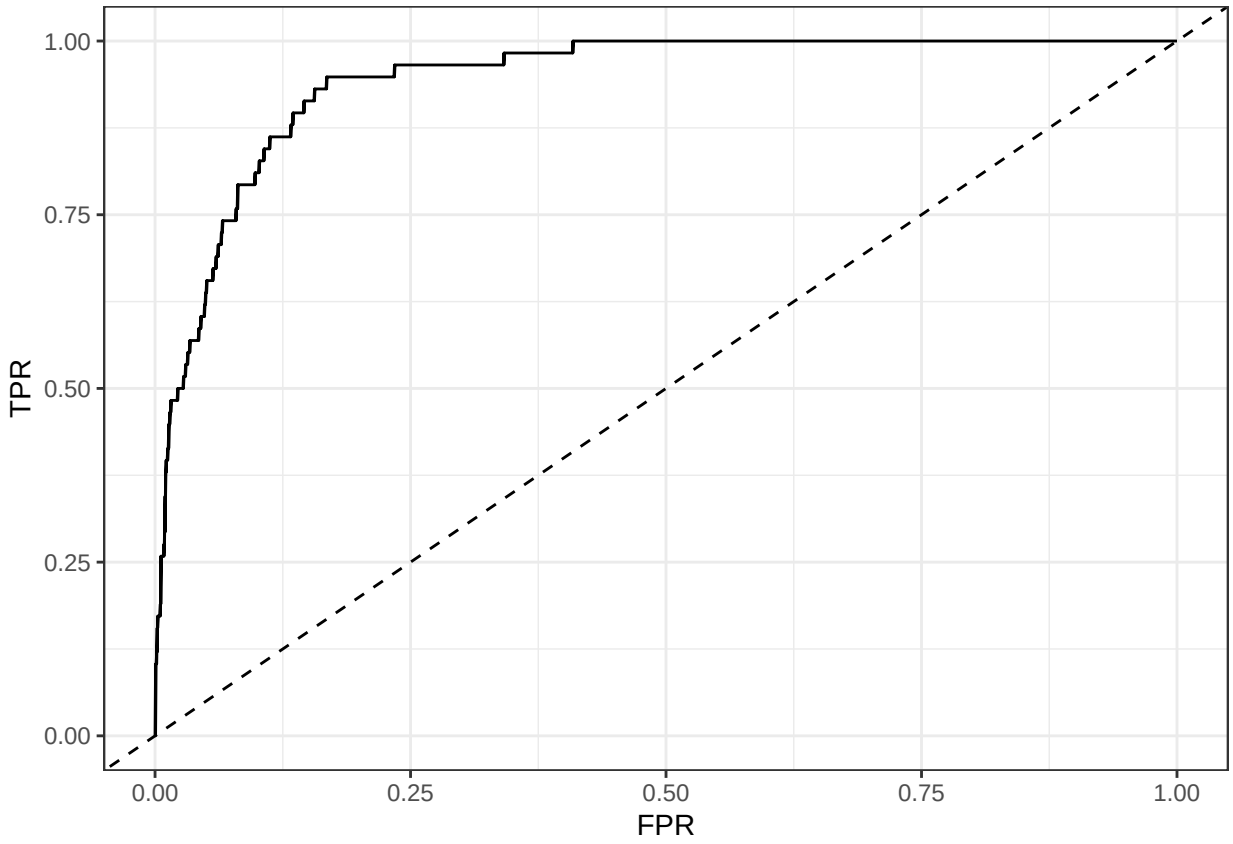
Next, let's plot the ROC curve for this classifier.

```
# 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()
```



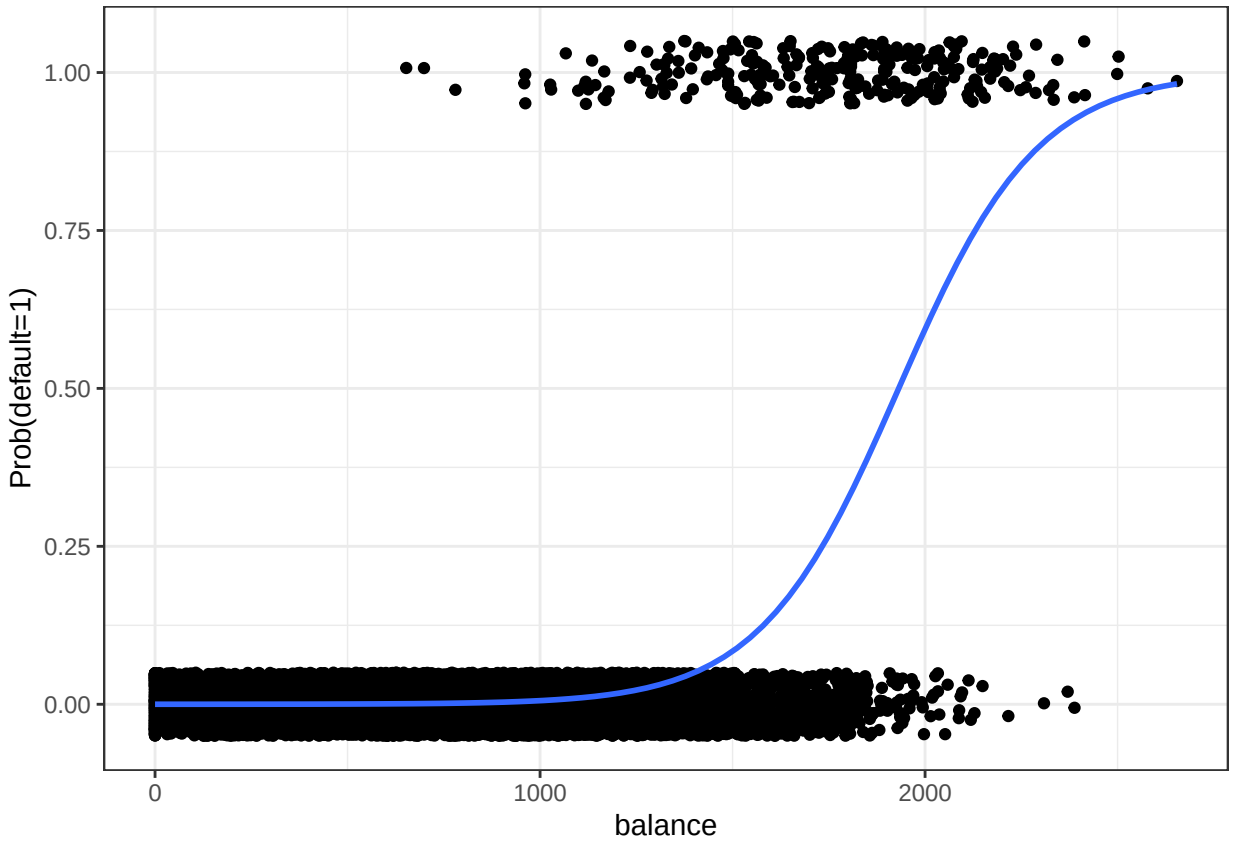

```
# print the AUC
roc_data$auc
```

```
## 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?

We switch from predicting no default to predicting default around balance = 2000.