Case Study: 2 or 7?

Machine Learning - Section 3.1.5

Marc Omar Haddad

Published: 23 February, 2020

Updated: 24 February, 2020

> *In the simple examples we've examined up to now, we only had one predictor. We actually do not consider these machine learning challenges, which are characterized by having many predictors.*
>
> — Dr. Rafael Irizarry

There are 784 predictors/features in a digit-reading machine learning algorithm. But for this case study we will select a subset of the aforementioned predictors. This subcategory will consist of **2 predictors** and **2 categories**.

Our goal for this lesson: Build an algorithm to determine if a digit is 2 or 7 (our 2 categories) from predictors. Our 2 predictors will be:

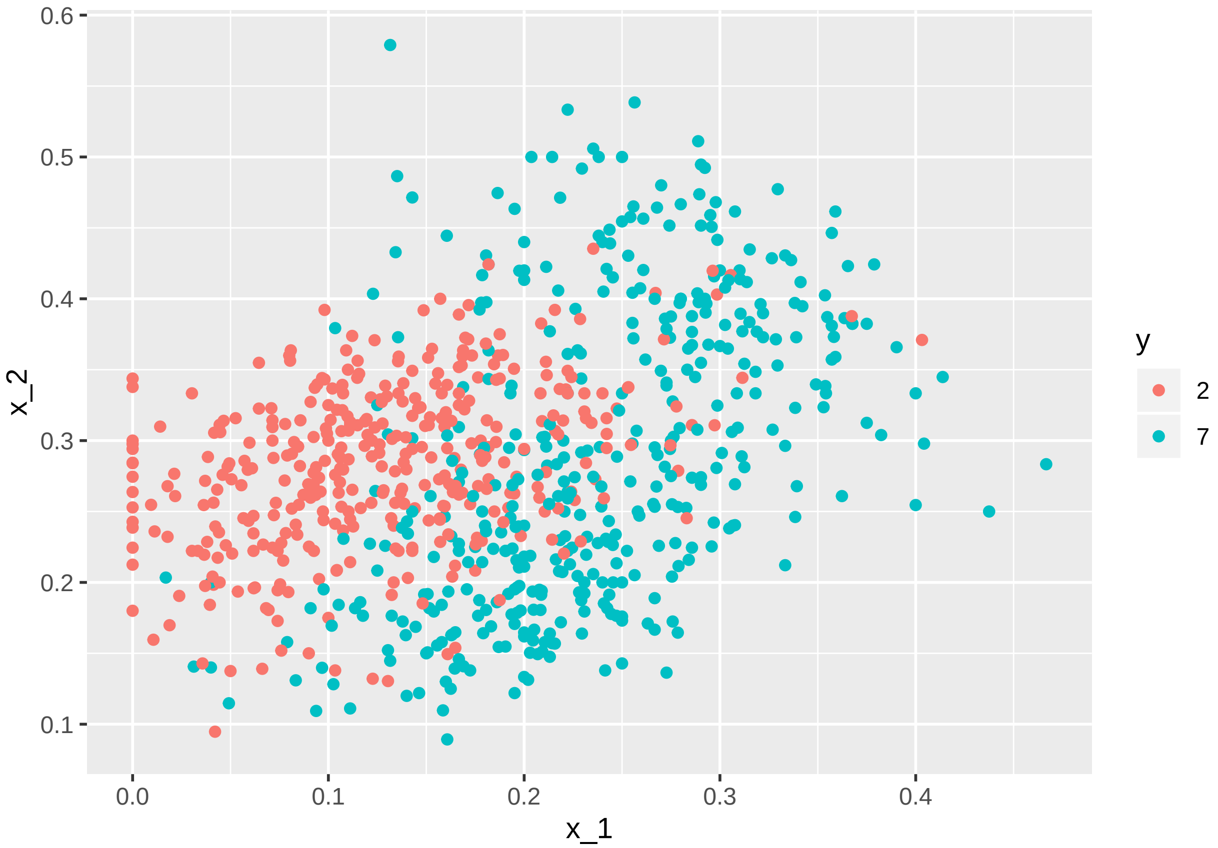
1. **The proportion of dark pixels in upper-left quadrant** X_1.
2. **The proportion of dark pixels in lower-right quadrant** X_2.

We will use a random sample of $1,000$ digits from the $60,000$ digits in our training set. 500 digits will be allocated to our custom **training set** and 500 digits will be allocated to our custom **test set**.
These sets can be accessed within the `dslabs` package by running `data("mnist_27")`.

First, we plot our two predictors from the data to explore it. We will use colors to denote labels.

```
data("mnist_27")

data = mnist_27$train %>% drop_na()
data %>% ggplot(aes(x_1, x_2, color = y)) + geom_point()
```



We can clearly see some patterns in the above plot:

- 7's are associated with large black values in the upper-left quadrant (approx. $X_1 >= 0.3$).
- 2's are associated with mid-range black values in the lower-right quadrant (approx. $0.2 < X_2 < 0.35$).

In this case study, we will attempt to fit a logistic regression model to our data.

Mathematically Expressing Logistic Regression

We will assume that "7" is the "positive" outcome (i.e. $Y = 1$).

Our model can be expressed mathematically like so:

$$\begin{aligned} g\{p(x_1, x_2)\} &= g\Big\{\Pr(Y = 1 \mid X_1 = x_1,\ X_2 = x_2)\Big\} \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \end{aligned}$$

Or, alternatively:

$$\begin{aligned} p(x_1, x_2) &= \Pr(Y = 1 \mid X_1 = x_1,\ X_2 = x_2) \\ &= g^{-1}(\beta_0 + \beta_1 x_1 + \beta_2 x_2) \end{aligned}$$

With $g^{-1}(x)$ being the inverse of the logistic function $g(x)$:

$$g(x) = \log \frac{x}{1 - x}$$

$$g^{-1}(x) = \frac{\exp(x)}{1 + \exp(x)}$$

In simple english,

First method:

- The **logistic transformation** ($\mathbf{g\{\}}$) of the **probability** of the **outcome being "7"** ($\mathbf{Y = 1}$) *given*:
  - The black value in the upper-left quadrant ($\mathbf{x_1}$) *and*
  - The black value in the lower-right quadrant ($\mathbf{x_2}$)
- Is *equal to* the **linear function** $\beta_0 + \beta_1 x_1 + \beta_2 x_2$

Alternative method:

- The **probability** of the **outcome being "7"** ($\mathbf{Y = 1}$) *given*:
  - The black value in the upper-left quadrant ($\mathbf{x_1}$) *and*
  - The black value in the lower-right quadrant ($\mathbf{x_2}$)
- Is *equal to* the **inverse logistic transformation** ($\mathbf{g^{-1}\{\}}$) of the **linear function** $\beta_0 + \beta_1 x_1 + \beta_2 x_2$

Fitting and Applying Logistic Regression

Despite the slight challenge involved in mathematically expressing logistic regression, fitting our model in `R` is fairly straight forward:

```r
fit = glm(y ~ x_1 + x_2, data = mnist_27$train, family = "binomial")
```

With our fit we can now build a **decision rule** based on the **estimate of the conditional probability**.

Decision Rule: If the estimate of the conditional probability is greater than 0.5, predict "7".

```r
p_hat = predict(fit, newdata = mnist_27$test)
y_hat = ifelse(p_hat > 0.5, 7, 2) %>% factor()

# Note: Our matrix indicates "2" as our positive class, but that's okay.
confusionMatrix(y_hat, mnist_27$test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  2  7
##          2 92 34
##          7 14 60
##
##                Accuracy : 0.76
##                  95% CI : (0.6947, 0.8174)
##     No Information Rate : 0.53
##     P-Value [Acc > NIR] : 1.668e-11
##
##                   Kappa : 0.5124
##
##  Mcnemar's Test P-Value : 0.006099
##
##             Sensitivity : 0.8679
##             Specificity : 0.6383
##          Pos Pred Value : 0.7302
##          Neg Pred Value : 0.8108
##              Prevalence : 0.5300
##          Detection Rate : 0.4600
##    Detection Prevalence : 0.6300
##       Balanced Accuracy : 0.7531
##
##        'Positive' Class : 2
##
```

As we can see in the `confusionMatrix()` results, we obtained an accuracy of 0.76.