# 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**.

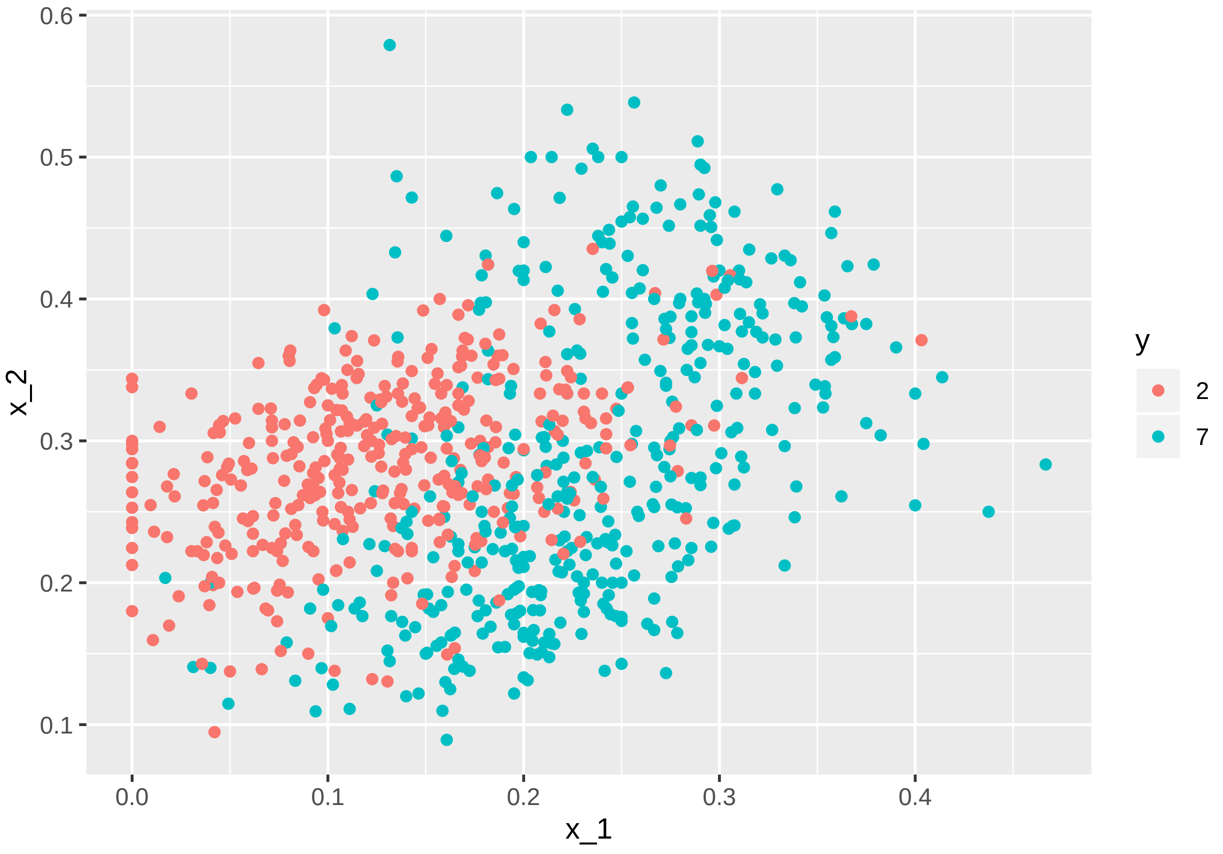
Our goal for this lesson: Build an algorithm to determine if a digit is 2 or 7 (our 2 categories) from predictors. Our 2 predictors will be:

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.

```r
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:

```
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.

Pretty good, but we can do better.

### Comparing Estimates with True Conditional Probabilities

### But First, Some Considerations

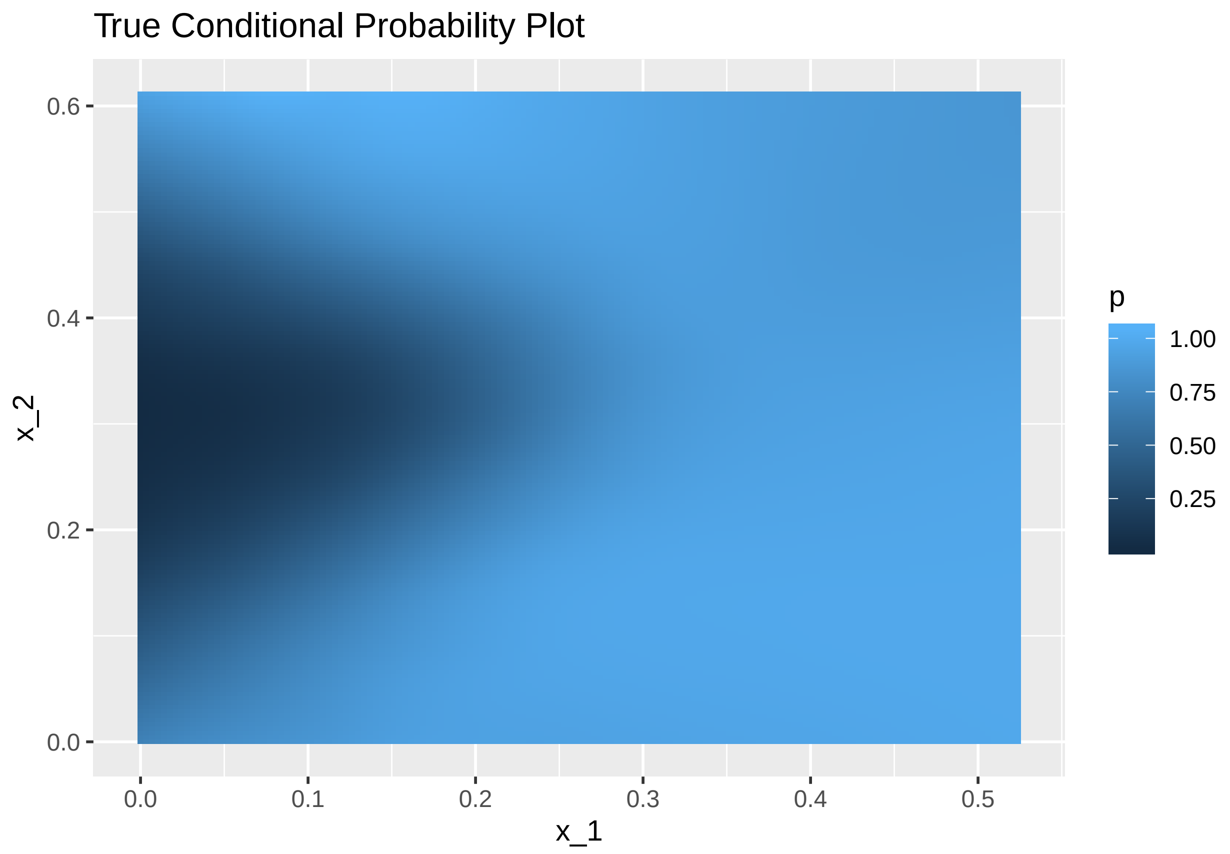
It must be noted, however, that contrary to data sets we use in practice, *this* particular data set's **true conditional probability is *known***. We know the true conditional probability because this example was constructed utilizing the entire set of $60,000$ digits. These $60,000$ digits were used to extract the true conditional probability $p(x_1, x_2)$.

The true conditional probability is included in our case study so we can **compare our estimates to it**, which allows us to assess different algorithms' limitations.

### Plotting the True Conditional Probability

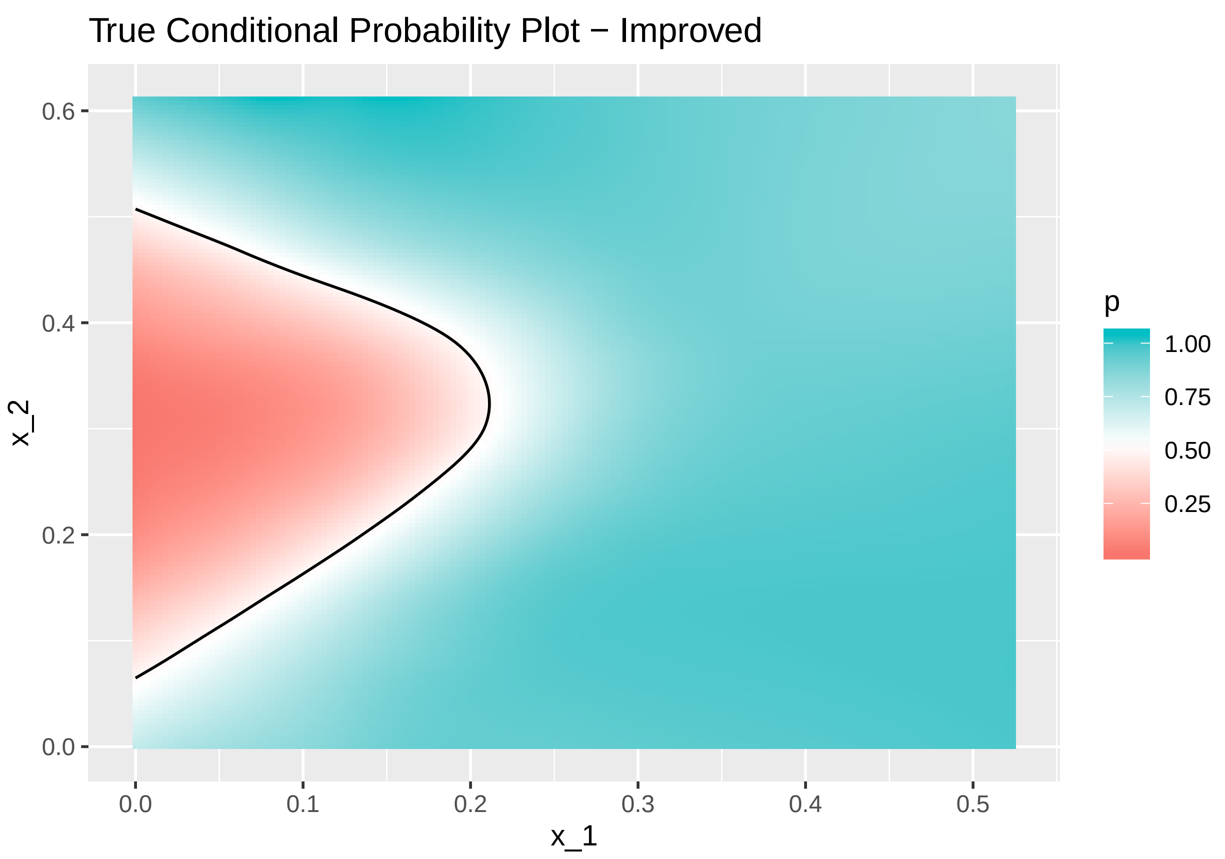
The `ggplot` function `geom_raster()` is a great tool to visualize our true conditional probability:

```
mnist_27$true_p %>% ggplot(aes(x_1, x_2, fill = p)) +
  geom_raster() +
  ggtitle("True Conditional Probability Plot")
```



To further improve this plot we will choose different colors and draw a line that separates the areas in which the conditional probabilities are higher and lower than 0.5:

```
mnist_27$true_p %>% ggplot(aes(x_1, x_2, z = p, fill = p)) +
  geom_raster() +
  scale_fill_gradientn(colors = c("#F8766D", "white", "#00BFC4")) +
  stat_contour(breaks = c(0.5), color = "black") +
  ggtitle("True Conditional Probability Plot - Improved")
```



**The Limitations of Logistic Regression**

The inflexibility of logistic regression is apparent when comparing its estimated conditional probability plot with the actual conditional probability plot.

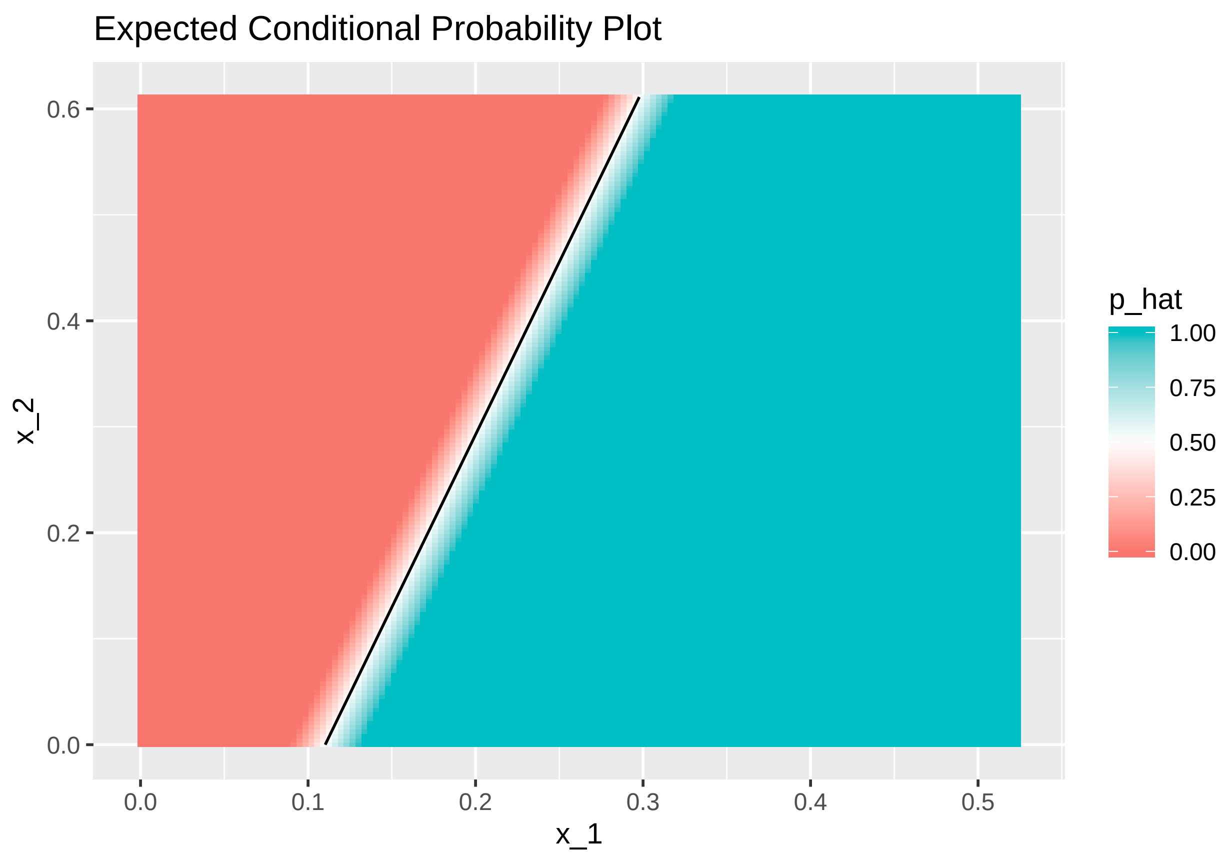
First, we must compute the boundary that divides the values of $x_1$ and $x_2$ that make the estimated conditional probability higher and lower than 0.5. In other words, our calculated boundary will reflect the conditional probability $\hat{p}(x_1, x_2)$ of being equal to 0.5.

Recall that our plot consists of $x_2$ as the **y-axis** and $x_1$ as the **x-axis**. So, in order to plot our logistic regression curve, we need to **express** $x_2$ **in terms of** $x_1$. Mathematically done like this:

$$p(x_1, x_2) = g^{-1}(\hat{\beta}_0+\hat{\beta}_1 x_1+\hat{\beta}_2 x_2) = 0.5 \implies \hat{\beta}_0+\hat{\beta}_1 x_1+\hat{\beta}_2 x_2 = g(0.5) = 0 \implies x_2 = \frac{-\hat{\beta}_0 - \hat{\beta}_1 x_1}{\hat{\beta}_2} = \frac{-\hat{\beta}_0}{\hat{\beta}_2} - \frac{\hat{\beta}_1}{\hat{\beta}_2} x_1$$

This equation derives the **y-intercept** $\frac{-\hat{\beta}_0}{\hat{\beta}_2}$, and the **slope** $-\frac{\hat{\beta}_1}{\hat{\beta}_2}$ of our logistic model.

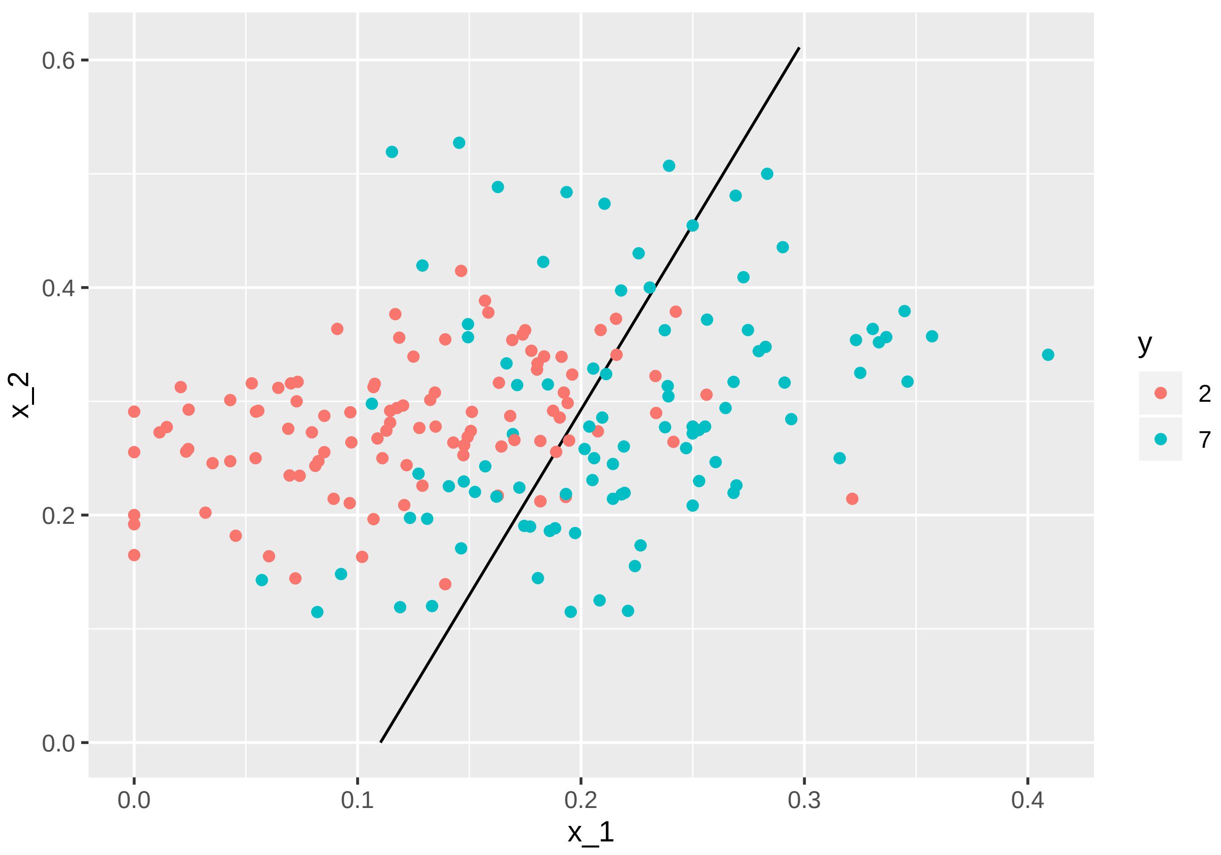
```
# Predict cond. prob. on newdata using earlier log fit
p_hat =  predict(fit, newdata = mnist_27$true_p)
p_hat = scales::squish(p_hat) # This constrains our estimates to be between 0 & 1

mnist_27$true_p %>%
  mutate(p_hat = p_hat) %>%
  ggplot(aes(x_1, x_2, z = p_hat, fill = p_hat)) +
  geom_raster() +
  scale_fill_gradientn(colors = c("#F8766D","white","#00BFC4")) +
  stat_contour(breaks = c(0.5), color = "black") +
  ggtitle("Expected Conditional Probability Plot")
```



By comparing the above plots, a clear issue arises: Due to its **linearity**, our logistic regression approach has *no chance* of predicting the **non-linear** nature of our true conditional probability.

Plotting our logistic regression against the earlier test data further reveals the above issue:

```
p_hat = predict(fit, mnist_27$true_p)
mnist_27$true_p %>%
  mutate(p_hat = p_hat) %>%
  ggplot() +
  stat_contour(aes(x_1, x_2, z = p_hat), breaks = c(0.5), color = "black") +
  geom_point(mapping = aes(x_1, x_2, color = y), data = mnist_27$test)
```



The bisecting line in the above plot miss-classifies several points due to the inflexibility of logistic regression in this case study.

Better, more flexible, alternatives to logistic regression include **Nearest Neighbor algorithms** and **Kernel approaches**.