Generative models

Jaime Davila

3/09/2022

Introduction

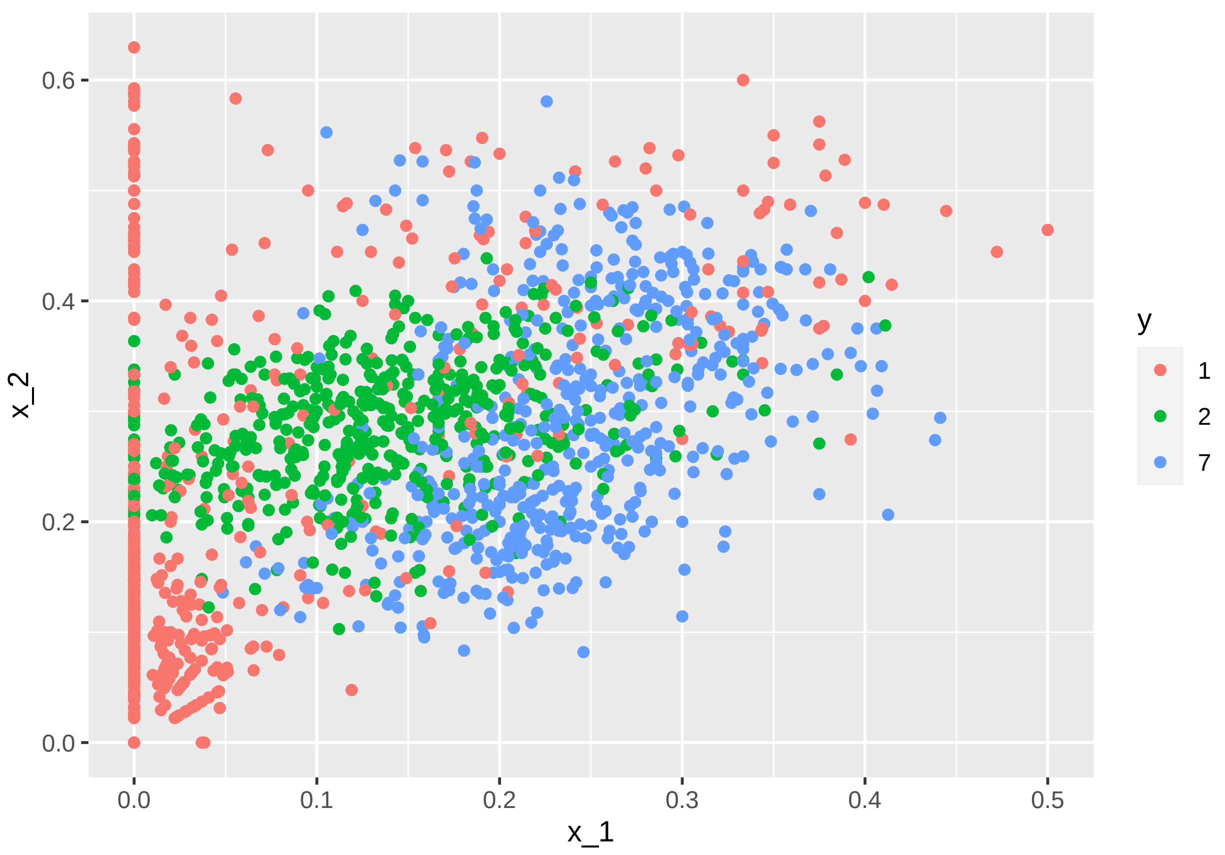
Our example of today is a variation of the `mnist_27` dataset, however this time we will add `1` to the mix. If you are interested in how this dataset was generated please read https://rafalab.github.io/dsbook/examples-of-algorithms.html#case-study-more-than-three-classes

Let's start by loading our testing and training datasets

```r
digits <- c("1","2","7")
train.127.tbl <- read_csv("~/Mscs 341 S22/Class/Data/train.127.csv") %>%
  mutate(y=factor(y, levels=digits))
test.127.tbl <- read_csv("~/Mscs 341 S22/Class/Data/test.127.csv") %>%
  mutate(y=factor(y, levels=digits))
```

And let's take a quick look at our training dataset

```r
ggplot(train.127.tbl, aes(x_1, x_2, color=y))+
  geom_point()
```



As you can see the problem that we are trying to solve is a classification problem where we have 3 potential values. The main idea of how to solve this particular problem is that we would need to calculate the probability of an observation been a `1`, the probability of been a `2` and the probability of being a `7` (notice this last probability is just 1 minus the other two values). Finally if we have to pick a label, we choose the one that maximizes the probability

Linear Discriminant Analysis

Suppose that we take the first coordinate (x) of an observation from our plot and we want to know the probability that such point is a `2`. We can write that using mathematical notation as:

$$P(Y = 2|X = x)$$

Using Bayes' theorem we know that:

$$P(Y = 2|X = x)) = \frac{P(X = x|Y = 2)P(Y = 2)}{P(X = x)}$$

Linear Discriminant Analysis works by assuming that the probability of having coordinate x for each digit follows a normal distribution and that the standard deviation of such distribution is the same for each our three digits (notice that the mean for 1 is a small value, the mean for 2 is in the middle and the mean for 7 is above 0.3). In other words,

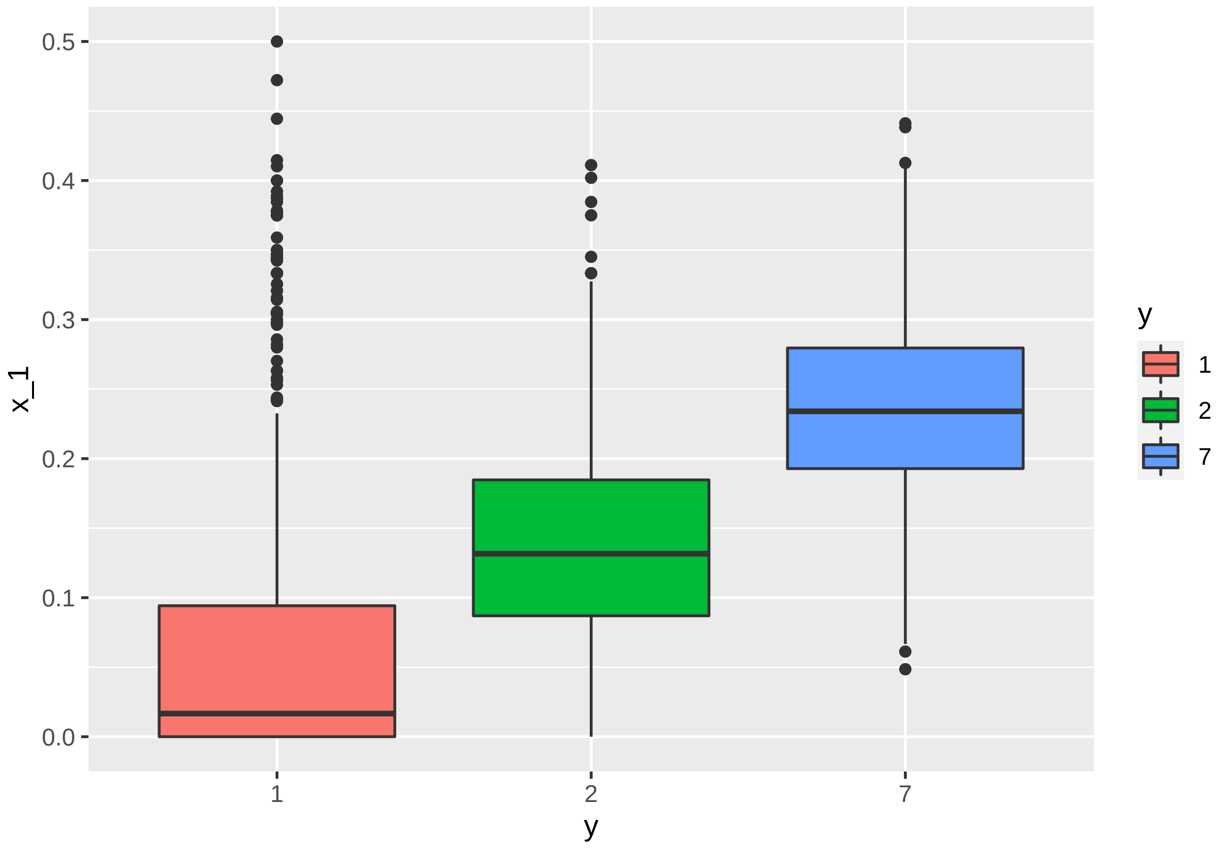
$$P(X = x|Y = k) = \frac{1}{\sqrt{2\pi}\sigma}e^{-x-\mu_k^2/(2\sigma^2)}$$

Notice that for this equation we need to calculate μ_1, μ_2, μ_7 and $sigma$. Using algebra we get that:

- μ_k is just the mean of x in each class k. $k = 1, 2, 7$.
- $\sigma^2 = \frac{1}{N-3} \sum_{k=1}^{3} (N_k - 1)\sigma_k^2$, where σ_k^2 is the variance of x in class k and N_k is the number of observation in class k

Before we proceed let's try to see to see the distribution of x_1 across our 3 categories of digits

```
ggplot(train.127.tbl, aes(x=y, y=x_1, fill=y))+
  geom_boxplot()
```



LDA in tidymodels

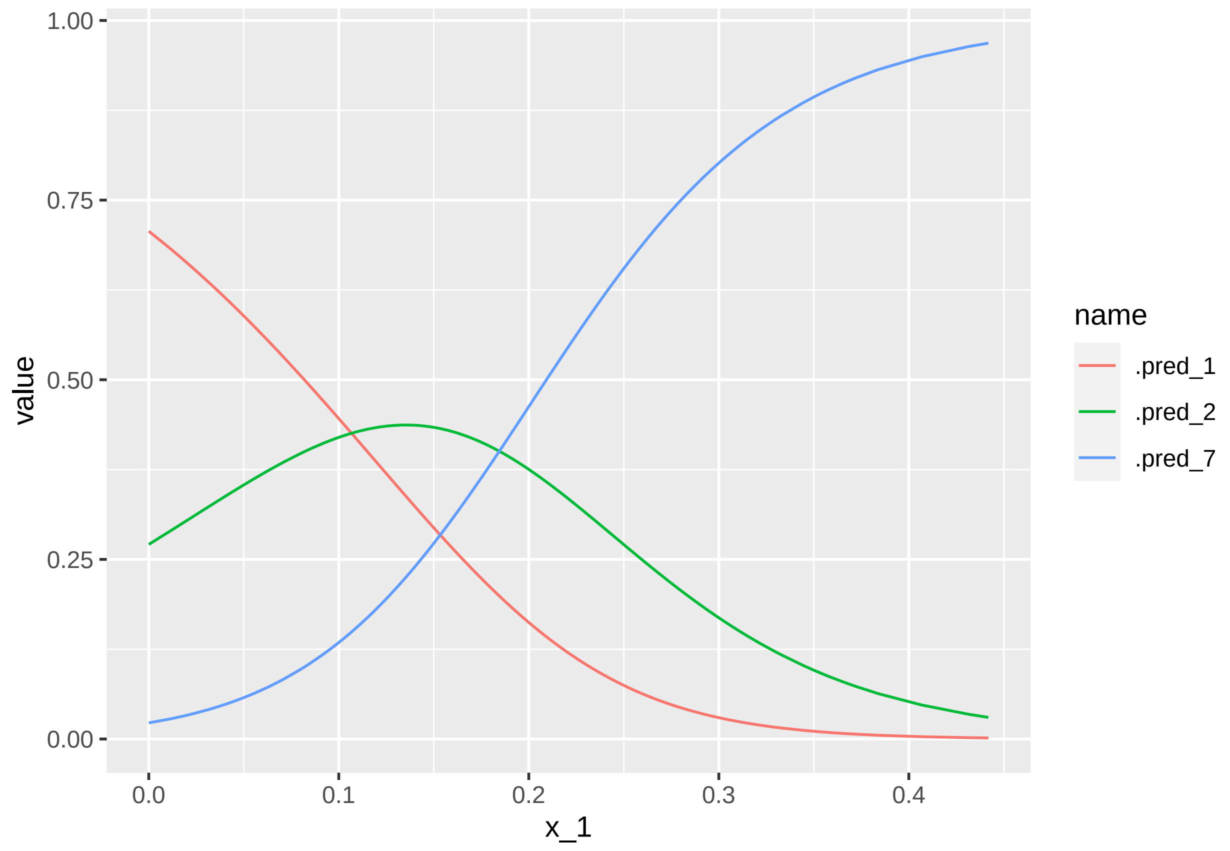
Let's start by setting up our LDA model using `tidymodels`. Notice that at this point we are only using feature x_1 to predict what type of digit we have

```
library(tidymodels)
library(discrim)
tidymodels_prefer()

lda.model <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

recipe <- recipe(y ~ x_1, data=train.127.tbl)
```

```
lda.wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(lda.model)

lda.fit <- fit(lda.wflow, train.127.tbl)
```

1. Use the command `augment` to add prediction information on your testing dataset. How many additional columns do you have? Plot the probabilities of being a `1`, `2` or `7` as function of `x_1`. Using the graph, what is the approximate intervals where observations are predicted to be 2s? How about 7?

```
augment(lda.fit, test.127.tbl) %>%
  pivot_longer(c(5:7)) %>%
  ggplot(aes(x_1,value,color=name)) +
  geom_line()
```



2. Calculate the accuracy of your method. Create the confusion matrix of this model using our testing dataset. What two pairs of digits are getting confused the most?

```
augment(lda.fit, test.127.tbl) %>%
  accuracy(truth = y, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.612
```

```
augment(lda.fit, test.127.tbl) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   1   2   7
##          1 100  44   5
##          2  16  48  33
##          7  26  31  96
```

3. Create an lda model that uses both x_1 and x_2 as input variables and calculate the accuracy of your method. Is this an improvement over 1)? Create the confusion matrix using our dataset and check what pairs of digits are getting missclassified the most.

```
recipe <- recipe(y ~ x_1+x_2, data=train.127.tbl)

lda.wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(lda.model)

lda.fit2 <- fit(lda.wflow, train.127.tbl)

augment(lda.fit2, test.127.tbl) %>%
  accuracy(truth = y, estimate = .pred_class)
```

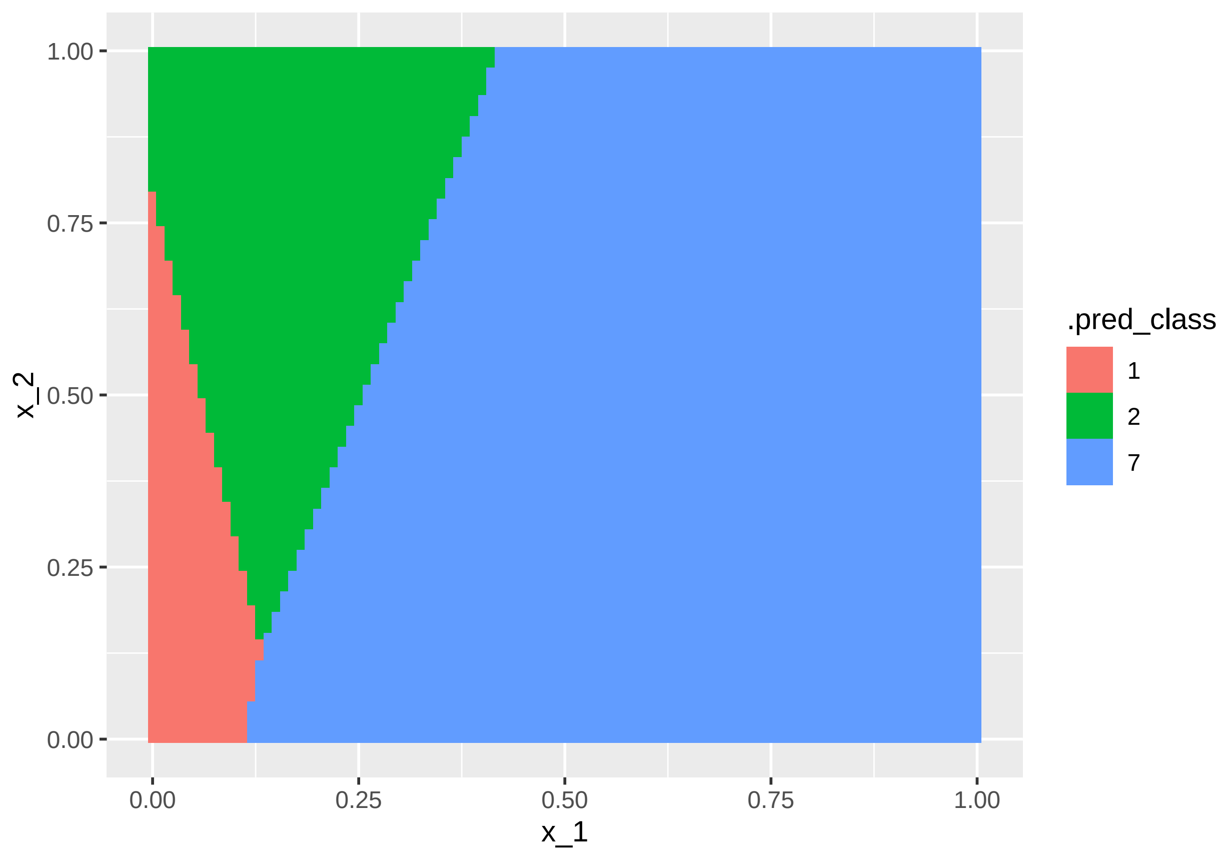
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.629
```

```
augment(lda.fit2, test.127.tbl) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   1   2   7
##          1 101  41   5
##          2  21  53  32
##          7  20  29  97
```

4. Create a grid with values in the unit square and plot the predicted class (using color) for each point in this grid.

```
create_grid <- function(delta) {
  grid.vec = seq(0,1, by=delta)
  expand_grid(x_1=grid.vec, x_2=grid.vec)
}
grid.tbl <- create_grid(0.01)

augment(lda.fit2, grid.tbl) %>%
  ggplot(aes(x_1, x_2, fill = .pred_class)) +
    geom_raster()
```



Quadratic discriminant analysis (QDA)

QDA uses similar ideas as LDA but the key distinction is that it *does not assume* an equal standard deviation structure across all the classes (see https://rafalab.github.io/dsbook/examples-of-algorithms.html#quadratic-discriminant-analysis for more details). QDA is readily available in `tidymodels`, just make sure in your model definition you use the function `discrim_quad()` instead of `discrim_linear()`

5. Calculate the confusion matrix and accuracy of QDA on the 3 digit problem using your testing dataset. Is the accuracy better when compared to LDA? Why do you think that is the case?

```
qda.model <- discrim_quad() %>%
  set_engine("MASS") %>%
  set_mode("classification")

recipe <- recipe(y ~ x_1+x_2, data=train.127.tbl)

qda.wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(qda.model)

qda.fit <- fit(qda.wflow, train.127.tbl)

augment(qda.fit, test.127.tbl) %>%
  accuracy(truth = y, estimate = .pred_class)
```

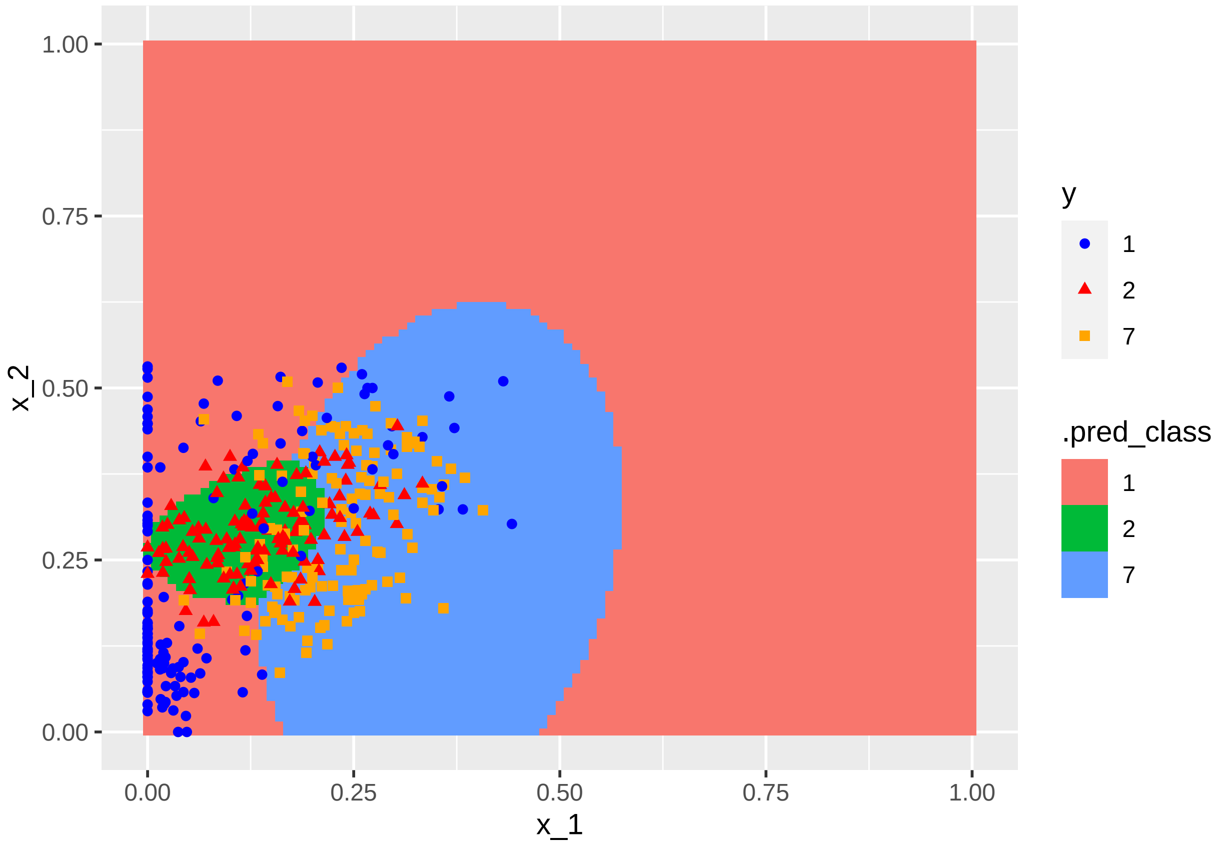
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.749
```

```
augment(qda.fit, test.127.tbl) %>%
  conf_mat(truth = y, estimate = .pred_class)
```

```
##           Truth
## Prediction   1   2   7
##          1 111   9  11
##          2  10  86  21
##          7  21  28 102
```

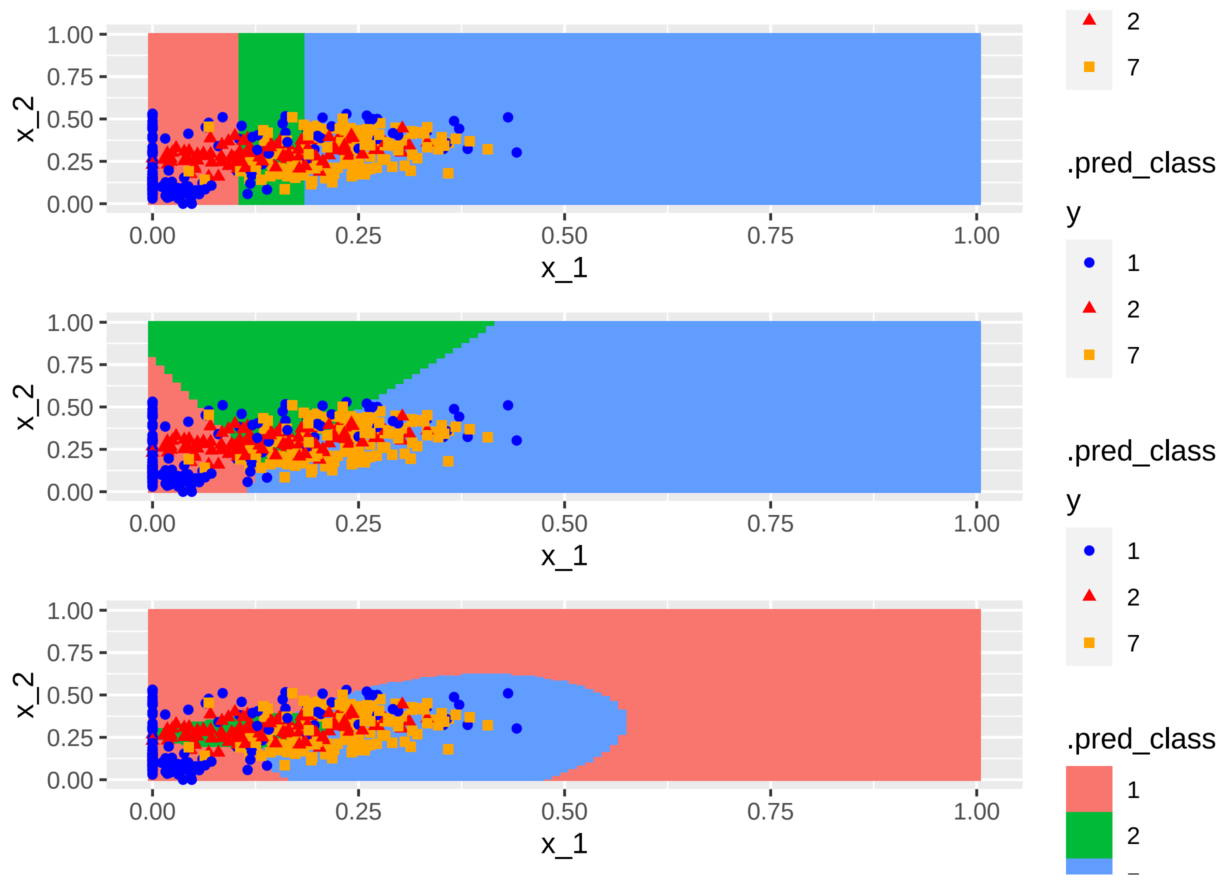
6. Create a grid with values in the unit square and plot the predicted class (using color) for the QDA model in each point in this grid. How does this compare to 4?

```
plot_boundary <- function(fit){
  augment(fit, create_grid(0.01))%>%
    ggplot() +
      geom_raster(aes(x_1, x_2, fill = .pred_class)) +
      geom_point(data=test.127.tbl, aes(x=x_1, y=x_2, color=y, shape=y))+
      scale_color_manual(values=c("blue","red","orange"))
}

plot_boundary(qda.fit)
```



And we can see the three models that we created in a single graph using `gridExtra`

```
library(gridExtra)

grid.arrange(plot_boundary(lda.fit),
             plot_boundary(lda.fit2),
             plot_boundary(qda.fit),
             nrow = 3)
```



Bonus

In practice it is useful to evaluate the performance of several methods at the same time. This is explored in more detail in: https://emilhvitfeldt.github.io/ISLR-tidymodels-labs/classification.html#extra---comparing-multiple-models

In this worksheet we have introduced so far 3 models:

- LDA with just x_1 (`lda.fit`)
- LDA with just x_1, x_2 (`lda.fit2`)
- QDA with just x_1, x_2 (`qda.fit`)

Our first step is to create a list that contains these 3 models as follows:

```
models.lst <- list("LDA.1var" = lda.fit,
                "LDA.2var" = lda.fit2,
                 "QDA" = qda.fit)
```

The next step is to use `imap_dfr()` from the `purr` package to apply `augment` of each model to the training dataset. Basically this creates a tibble with the predictions of each model in a single place. We can do that as follows:

```
preds.tbl <- imap_dfr(models.lst, augment,
                  new_data = test.127.tbl, .id = "model")
preds.tbl
```

```
## # A tibble: 1,197 x 8
##    model    y        x_1    x_2 .pred_class .pred_1 .pred_2 .pred_7
##    <chr>    <fct>  <dbl>  <dbl> <fct>         <dbl>   <dbl>   <dbl>
##  1 LDA.1var 1     0      0.172  1             0.707   0.271   0.0225
##  2 LDA.1var 2     0.117  0.3    2             0.395   0.432   0.173
##  3 LDA.1var 1     0      0.3    1             0.707   0.271   0.0225
##  4 LDA.1var 1     0.353  0.324  7             0.0101  0.0925  0.897
##  5 LDA.1var 1     0.04   0.08   1             0.615   0.337   0.0480
##  6 LDA.1var 1     0.0682 0.477  1             0.539   0.381   0.0794
##  7 LDA.1var 1     0.298  0.316  7             0.0307  0.172   0.797
##  8 LDA.1var 1     0.0286 0.0857 1             0.643   0.318   0.0388
##  9 LDA.1var 2     0.245  0.391  7             0.0818  0.283   0.636
## 10 LDA.1var 1     0      0.469  1             0.707   0.271   0.0225
## # ... with 1,187 more rows
```

Notice how there is an additional column `model` in our `preds.tbl`.

We can add multiple different performance metrics using `metric_set` from `yardstick`. In our example we will use accuracy, sensitivity and specificity

```
multi.metric <- metric_set(accuracy, sensitivity, specificity)
```

And finally we can apply all these performance metrics across models using `group_by(model)` as follows

```
preds.tbl %>%
  group_by(model) %>%
  multi.metric(truth = y, estimate = .pred_class) %>%
  select(model, .metric, .estimate) %>%
  pivot_wider(names_from=.metric, values_from=.estimate)
```

```
## # A tibble: 3 x 4
##   model    accuracy sensitivity specificity
##   <chr>       <dbl>       <dbl>       <dbl>
## 1 LDA.1var    0.612       0.604       0.806
## 2 LDA.2var    0.629       0.622       0.815
## 3 QDA         0.749       0.747       0.875
```

No homework for next week, but remember your challenge is due next Wednesday 3/16!