Classification-2

Ivana K.

2/23/2022

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(dslabs)
```

Is it a 2 or 7? - continuation

Last time we started considering the problem of labeling digits as 2 or a 7 using the following variables (features):

- `x_1` will be the proportion of dark pixels in the upper left quadrant.
- `x_2` will be the proportion of dark pixels in the lower right quadrant.

Let's start by loading the dataset `mnist_27` from `dslabs` and creating our testing and training datasets:

```
data("mnist_27")
mnist.train.tbl <- tibble(mnist_27$train)
mnist.test.tbl <- tibble(mnist_27$test)
```

And let's note the dimensions of those datasets

```
dim(mnist.train.tbl)
```

```
## [1] 800   3
```

```
dim(mnist.test.tbl)
```

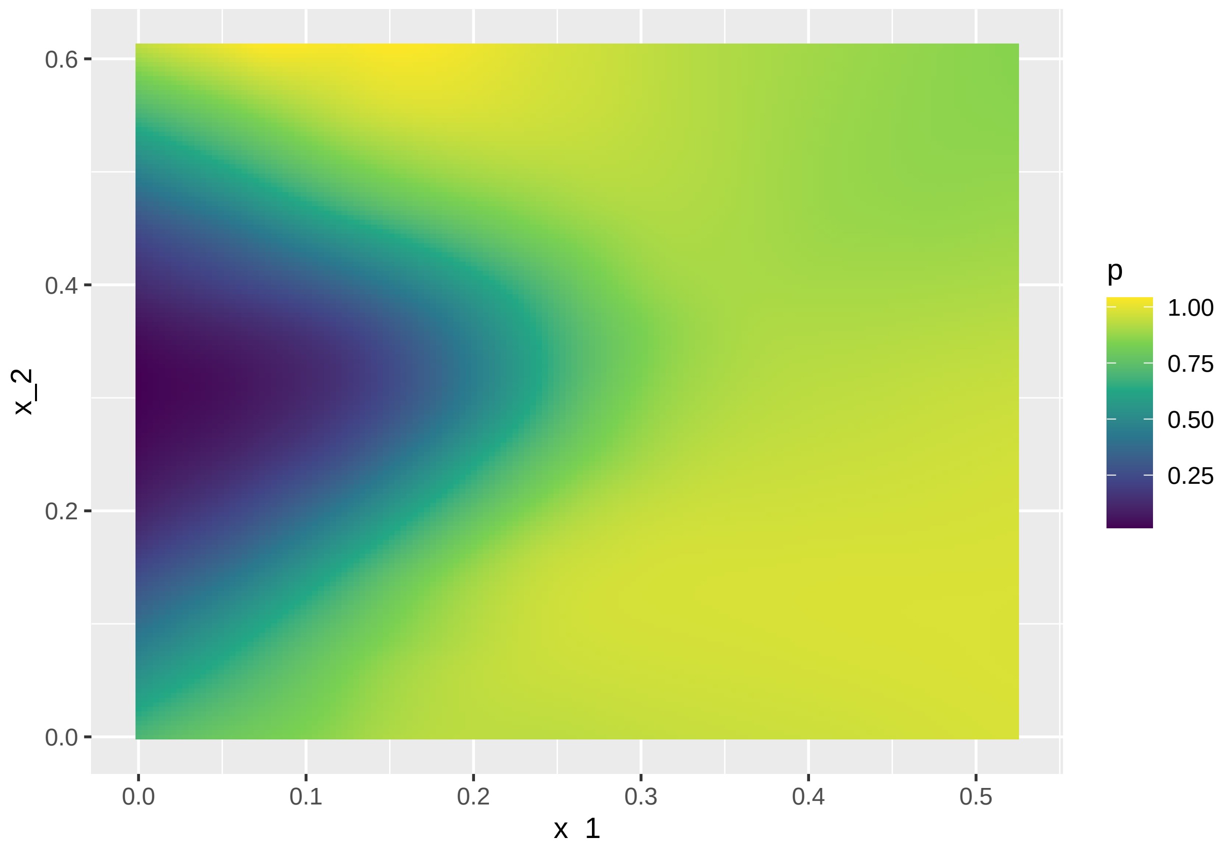
```
## [1] 200   3
```

Today we are interested in defining the *decision boundary* of the best theoretical classifier which we will call the *Bayes' boundary*. The `mnist` dataset has over 60,000 digits so we can approximate the theoretical probability of a 7 (compared to a 2). Luckily for us this information is contained in the field `true_p` of `mnist_27`. Let's take a look at it:

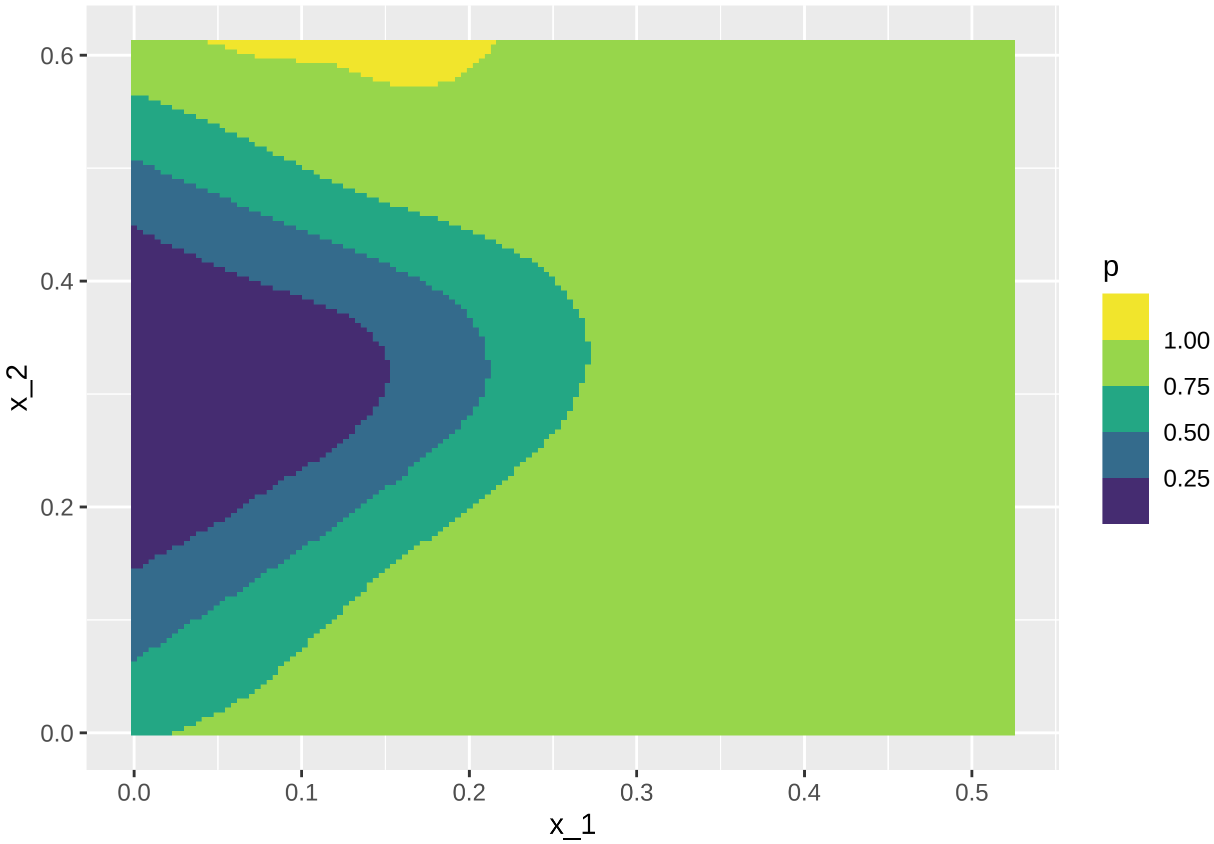
```
mnist.true.tbl <-  tibble(mnist_27$true_p)
```

The way to interpret this table is to note that given `x_1` and `x_2` it provides an estimate of the probability of a digit been a 7. Let's plot how this probability looks like in two similar ways

```
ggplot(mnist.true.tbl, aes(x_1, x_2, fill = p)) +
  geom_raster() +
  scale_fill_viridis_c()
```

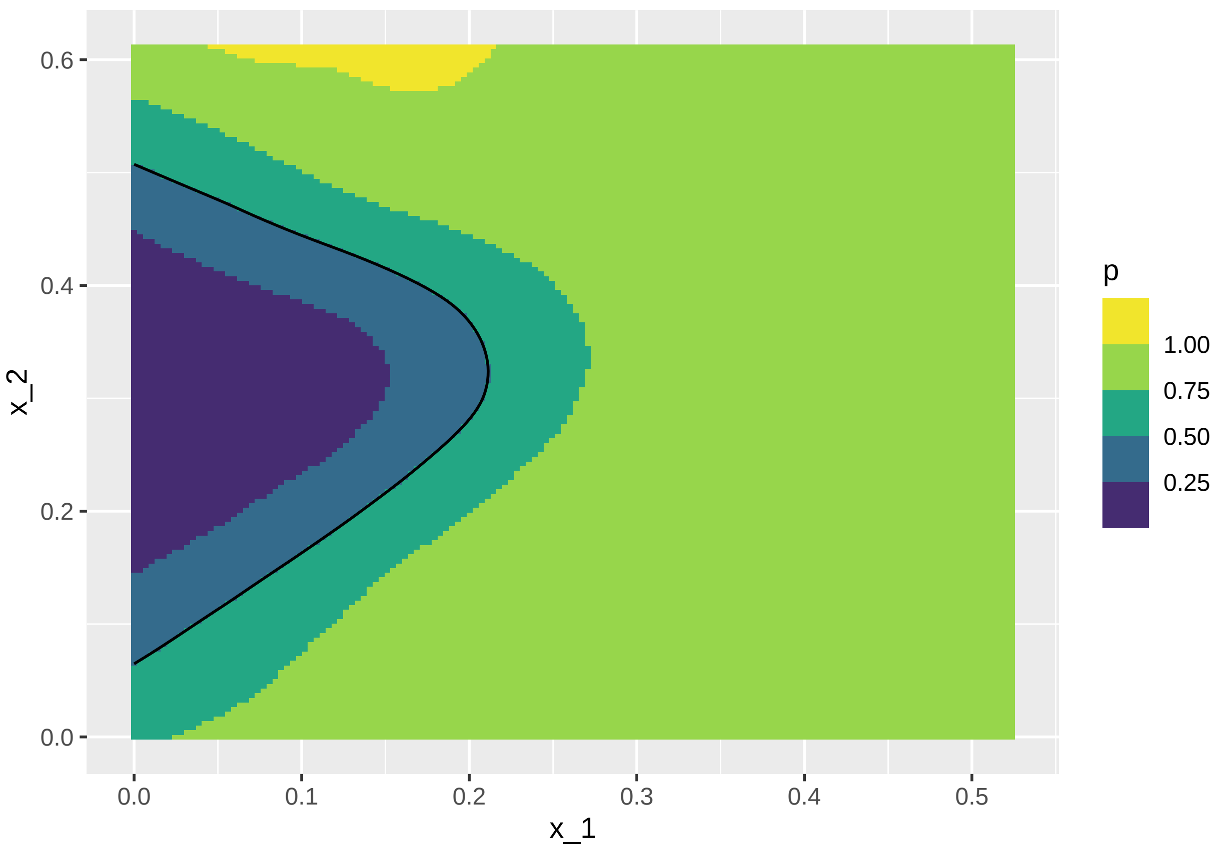



```
ggplot(mnist.true.tbl, aes(x_1, x_2, fill = p)) +
  geom_raster() +
  scale_fill_viridis_b()
```



Notice how points on the far right are likely to be `7`, and how points in the left are very likely to be `2`. Finally notice the probability changes around a curved region in the left of the screen.

The Bayes' boundary consists of all the points where the probability is exactly equal to 0.5. We can plot this boundary by using the `stat_contour` command of `ggplot`. Notice that for `stat_contour` to work you need to define `z` in your `aes` command:




```
## # A tibble: 22,500 x 3
##        x_1   x_2     p
##      <dbl> <dbl> <dbl>
##  1 0           0 0.703
##  2 0.00352     0 0.711
##  3 0.00703     0 0.719
##  4 0.0105      0 0.727
##  5 0.0141      0 0.734
##  6 0.0176      0 0.741
##  7 0.0211      0 0.747
##  8 0.0246      0 0.753
##  9 0.0281      0 0.759
## 10 0.0316      0 0.765
## # ... with 22,490 more rows
```

In the following exercises we will explore the decision boundary generated by our KNN classifier using the following steps:

1. Using a value of `kNear` of 10, create a KNN model using your training dataset

```
kNear=10
knn.model <- knn3(y~x_1+x_2, data=mnist.train.tbl, k=kNear)
```

2. We would like to visualize the values of our KNN model across all of the points of the unit square. However our testing dataset does not contain enough of those points so we need to create a tibble with a big amount of points from the unit square interval. We will do that in the following steps

a. Create a vector `grid.vec` that contains the numbers 0, 0.1, . . . ,1. Make use of the function `seq`.

```r
(grid.vec = seq(0,1, by=0.1))
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

b. Look at the documentation of the `expand_grid` function from the `tidyverse`, and create a tibble `grid.tbl` with two columns, `x_1` and `x_2` which contains a grid of combinations of two points from 0 to 1 by steps of 0.1

```r
(grid.tbl <- expand_grid(x_1=grid.vec, x_2=grid.vec))
```

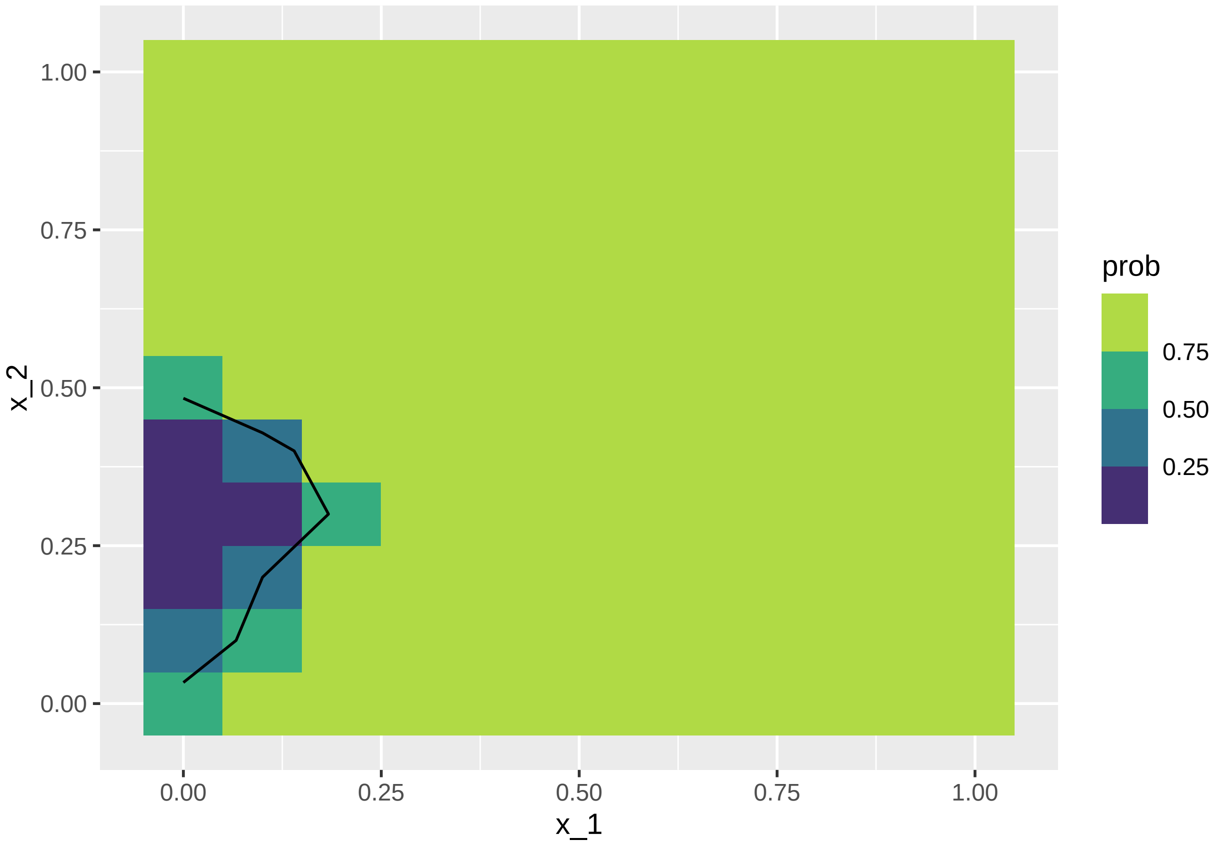
```
## # A tibble: 121 x 2
##      x_1   x_2
##    <dbl> <dbl>
##  1     0   0
##  2     0   0.1
##  3     0   0.2
##  4     0   0.3
##  5     0   0.4
##  6     0   0.5
##  7     0   0.6
##  8     0   0.7
##  9     0   0.8
## 10     0   0.9
## # ... with 111 more rows
```

c. Evaluate your KNN model on the values of `grid.tbl`. Create a new column `prob` in `grid.tbl` with the predicted probability of being a 7.

```r
pred <- predict(knn.model, grid.tbl)
grid.tbl <- grid.tbl %>%
  mutate(prob = pred[,2])
```

d. Use `grid.tbl` to plot the predicted probability across the unit grid and plot the *decision boundary*.

```r
ggplot(grid.tbl, aes(x_1, x_2, z=prob,fill = prob)) +
  geom_raster() +
  stat_contour(breaks=c(0.5), color="black")+
  scale_fill_viridis_b()
```

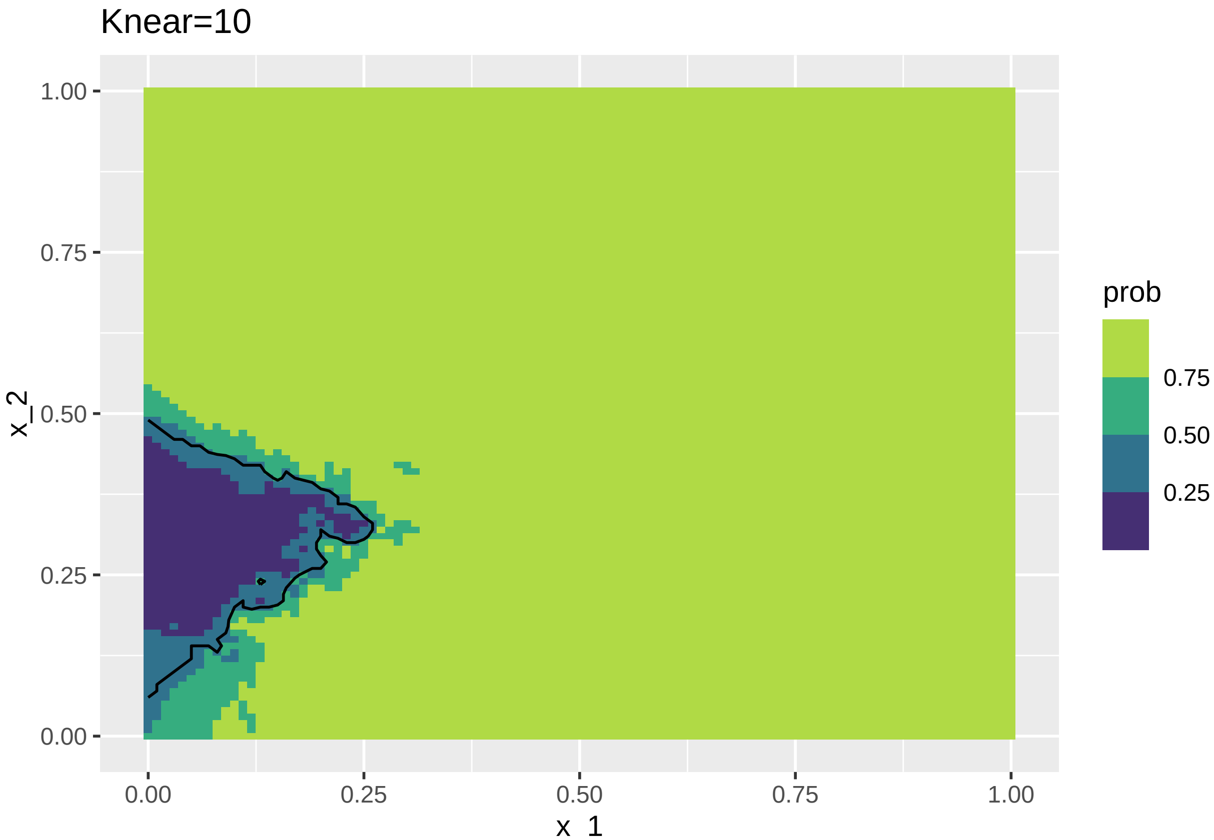


e. It seems your graph is too pixelated. Create a function `plot_knn_model(kNear, grid.dist, train.tbl)` that trains a KNN model with parameter `kNear` on `train.tbl` and displays the value of the probability of being a 7 on a grid of points generated every `grid.dist`. Evaluate your function using `grid.dist` equals to 0.1, 0.03 and 0.01

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

plot_knn_model <-  function (kNear, grid.dist, train.tbl) {
  knn.model <- knn3(y~x_1+x_2, data=train.tbl, k=kNear)

  grid.tbl <-  create_grid(grid.dist)
  pred <- predict(knn.model, grid.tbl)
  grid.tbl <- grid.tbl %>%
    mutate(prob = pred[,2])

  ggplot(grid.tbl, aes(x_1, x_2, z=prob,fill = prob)) +
    geom_raster() +
    stat_contour(breaks=c(0.5), color="black")+
    scale_fill_viridis_b()+
    ggtitle(str_c("Knear=",kNear))
}
```

```
#plot_knn_model(10,0.1, mnist.train.tbl)
#plot_knn_model(10,0.03, mnist.train.tbl)
```

```r
plot_knn_model(10,0.01, mnist.train.tbl)
```



3. Experiment plotting with different values of k (say from k=5 to k=50, using steps of 5). Which decision boundary looks more similar to the Bayes boundary? Is this consistent with the optimal value of `k` that you found using in point 4 of our last activity, `3_Classification.Rmd`?

plot_knn_model(5,0.01, mnist.train.tbl) plot_knn_model(10,0.01, mnist.train.tbl) plot_knn_model(15,0.01, mnist.train.tbl) plot_knn_model(20,0.01, mnist.train.tbl) plot_knn_model(25,0.01, mnist.train.tbl) plot_knn_model(30,0.01, mnist.train.tbl) plot_knn_model(35,0.01, mnist.train.tbl) plot_knn_model(40,0.01, mnist.train.tbl) plot_knn_model(45,0.01, mnist.train.tbl) plot_knn_model(50,0.01, mnist.train.tbl)

#{r include=FALSE} for (i in seq(5,150,10)){ p <- plot_knn_model(i,0.01, mnist.train.tbl) print(p) }

Decision boundary of a linear classifer

4. We can also use a linear model to approximate the probability of being a `7`. Notice that in order to make this approach work, we need to create a new input variable where `2`s are encoded as zeros and `7`s are encoded as ones. Also note that the linear model can give values outside of $[0, 1]$ so we will need to truncate predictions that are negative to 0 and prediction over 1 to 1. Implement this approach and plot the boundary of this classifier. How does this boundary compare to the boundary generated by the KNN model?

The boundary for the linear model is very straight and regular. The KNN model is very uneven. The linear graph shows that there is a higher probability of a number being 7 if it has fewer pigments in x_2.

The KNN graph shows more nuance than this. There is a much smaller region where the probability of the number being a 7 is less than 50%.

```
test <- mnist.train.tbl %>%
  mutate(y_1 = ifelse(y == "2", 0, 1))
linear.model <- lm(y_1~x_1+x_2, data = test)
#linear.model
```

```
(grid2.vec = seq(0,1, by=0.01))
```

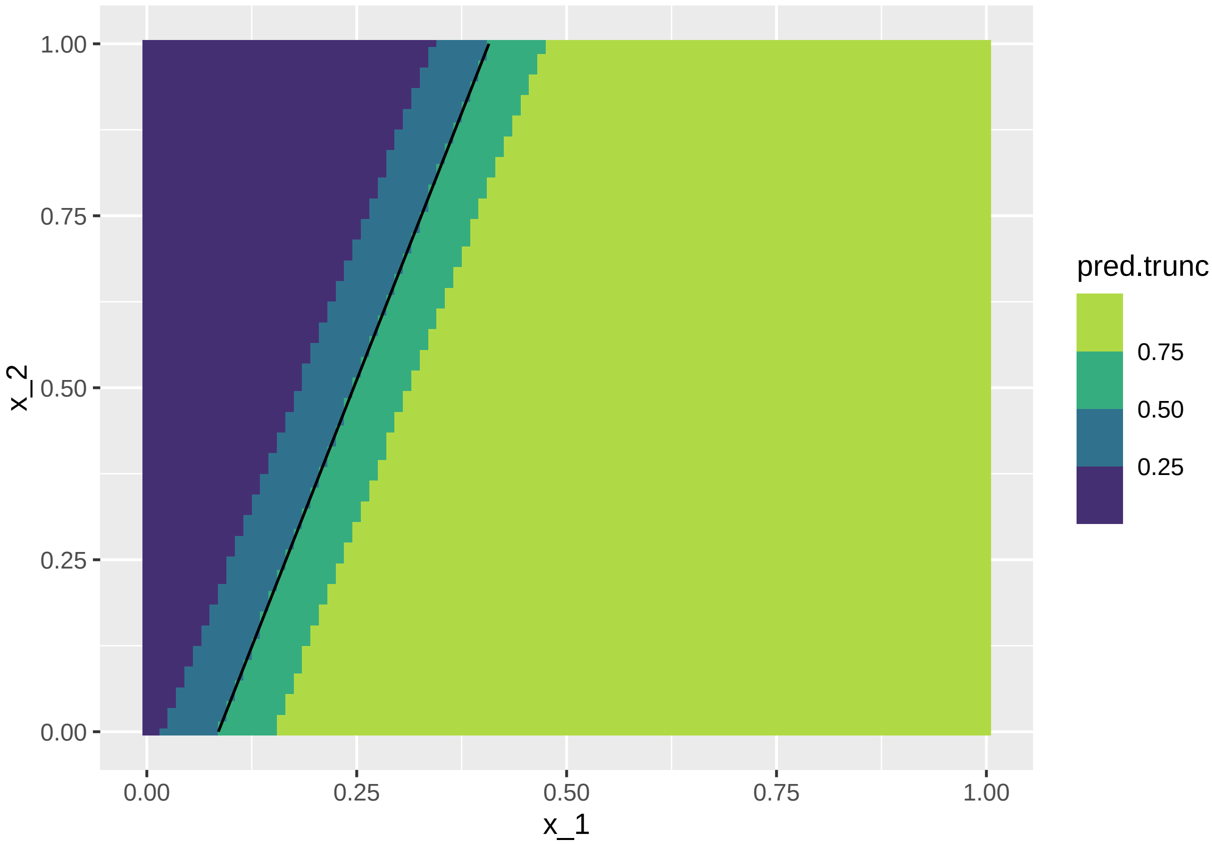
```
##   [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14
##  [16] 0.15 0.16 0.17 0.18 0.19 0.20 0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29
##  [31] 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40 0.41 0.42 0.43 0.44
##  [46] 0.45 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59
##  [61] 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70 0.71 0.72 0.73 0.74
##  [76] 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89
##  [91] 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.00
```

```
(grid2.tbl <- expand_grid(x_1=grid2.vec, x_2=grid2.vec))
```

```
## # A tibble: 10,201 x 2
##      x_1   x_2
##    <dbl> <dbl>
##  1     0  0
##  2     0  0.01
##  3     0  0.02
##  4     0  0.03
##  5     0  0.04
##  6     0  0.05
##  7     0  0.06
##  8     0  0.07
##  9     0  0.08
## 10     0  0.09
## # ... with 10,191 more rows
```

```
pred2 <- predict(linear.model, grid2.tbl)
grid2.tbl <- grid2.tbl %>%
  mutate(lm = pred2) %>%
  mutate(pred.trunc = ifelse(pred2 < 0, 0, ifelse(pred2 > 1, 1, pred2)))
```

```
ggplot(grid2.tbl, aes(x_1, x_2, z=pred.trunc,fill = pred.trunc)) +
  geom_raster() +
  stat_contour(breaks=c(0.5), color="black")+
  scale_fill_viridis_b()
```



The Default dataset

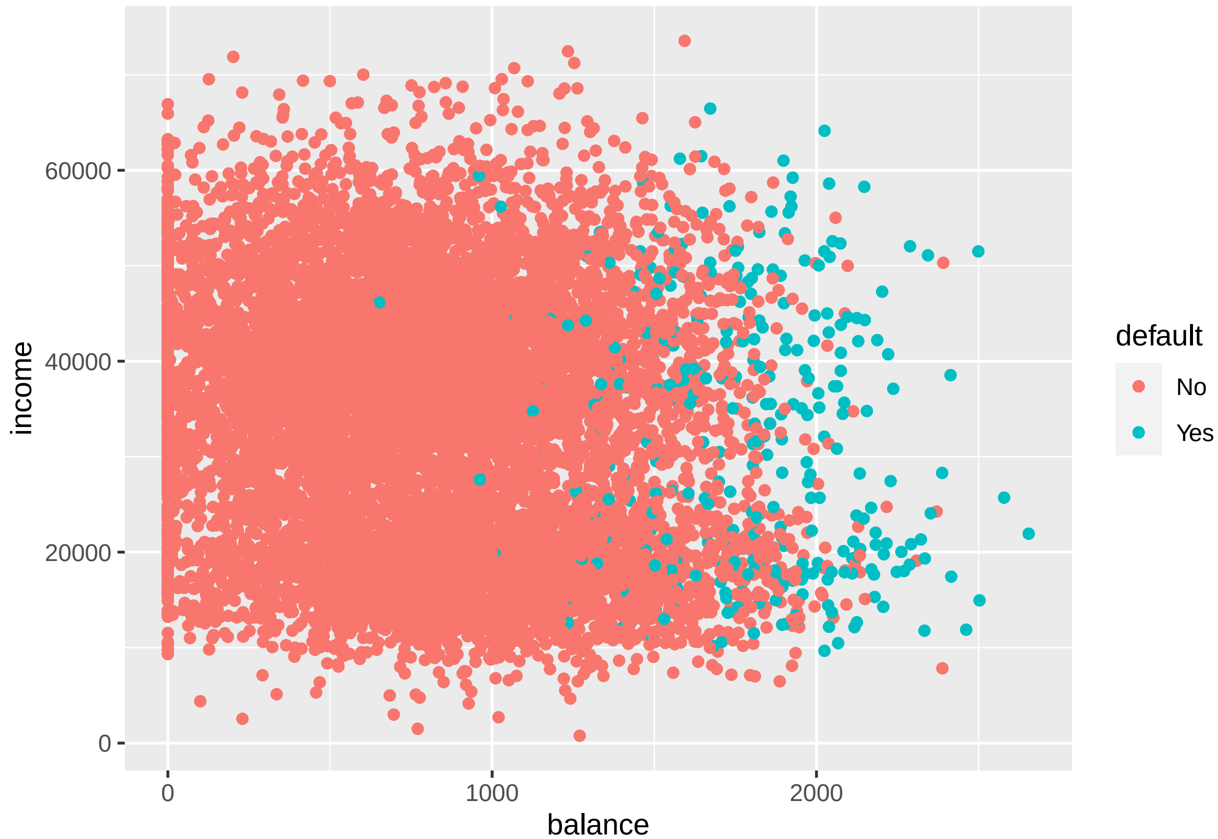
In the following sets of exercises we will be exploring the `Default` dataset available from the `ISLR2` package. In particular we will construct linear models that would allow us to predict whether a particular person would go on default.

```
library(ISLR2)
data(Default)
default.tbl <- tibble(Default)
default.tbl
```

```
## # 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
```

5. a. Generate a plot of balance (x) and income (y) vs default (using color). What trends do you observe?

```
ggplot(default.tbl, aes(x = balance, y = income, color = default)) +
  geom_point() +
  geom_jitter()
```



People with the highest balances are more likely to "go on default". For fun, I also did facet_wrap by student. The area of the graphs where people default is almost the same regardless of student status. Defaulting appears to be much more related to balance. (hmm. would the best linear model look at just balance?) Balance and income do not appear strongly related to one another.

b. Divide the original datasets into a training (8000 elements) and a testing dataset (2000 elements) by

```
set.seed(12345)
trainer <- slice_sample(default.tbl, n = 8000)
tester <- setdiff(default.tbl, trainer)
tester
```

```
## # A tibble: 2,000 x 4
##    default student balance income
##    <fct>   <fct>     <dbl>  <dbl>
##  1 No      Yes        920.  7492.
##  2 No      Yes          0  21871.
##  3 No      Yes       1221. 13269.
##  4 No      No         237. 28252.
##  5 No      No         286. 45042.
##  6 No      No           0  50265.
##  7 No      Yes        528. 17637.
```

```
## 8 No      No        1095. 26465.
## 9 No      No         643. 41474.
## 10 No     No         495. 54385.
## # ... with 1,990 more rows
```

6. Create a linear model (similar to point 4) that predicts `default` based on `balance` and `income`. Notice that the value of the model is not necessarily between 0 and 1, so divide the prediction by the maximum model value (across the grid) and truncate your response to 0 if the value is negative. What is the missclassification rate?

```r
trainer <- trainer %>%
  mutate(default_1 = ifelse(default == "No", 0, 1))

linear.model2 <- lm(default_1~balance+income, data = trainer)#+income
linear.model2
```

```
##
## Call:
## lm(formula = default_1 ~ balance + income, data = trainer)
##
## Coefficients:
## (Intercept)      balance       income
##  -9.491e-02    1.284e-04    5.988e-07
```

```r
pred3 <- predict(linear.model2, tester)
value <- max(pred3)
tester2 <- tester %>%
  mutate(lm = pred3/value) %>%
  mutate(lm = ifelse(pred3 < 0, 0, pred3)) %>%
  mutate(p = ifelse(lm < .5, "No", "Yes"))
#max "maximum model value" is pred3? 0.2379911
tester2
```

```
## # A tibble: 2,000 x 6
##    default student balance income      lm p
##    <fct>   <fct>     <dbl>  <dbl>   <dbl> <chr>
##  1 No      Yes        920.  7492. 0.0276  No
##  2 No      Yes          0  21871. 0       No
##  3 No      Yes       1221. 13269. 0.0697  No
##  4 No      No         237. 28252. 0       No
##  5 No      No         286. 45042. 0       No
##  6 No      No           0  50265. 0       No
##  7 No      Yes        528. 17637. 0       No
##  8 No      No        1095. 26465. 0.0615  No
##  9 No      No         643. 41474. 0.0125  No
## 10 No      No         495. 54385. 0.00118 No
## # ... with 1,990 more rows
```

misclassification rate:

```r
mean(tester2$default != tester2$p)
```

```
## [1] 0.036
```

[1] 0.036

7. Plot the probability of the model created on 6) on a grid where $(x_1, x_2) \in [0, 3000] \times [0, 80000]$. Make sure your grid **does not have** over 10,000 points. Plot the decision boundary of the model as well.

```r
grid.vec.3k = seq(0,3000, by=100)
grid.vec.80k = seq(0,80000, by=1000)
(grid.tbl.7 <- expand_grid(balance=grid.vec.3k, income=grid.vec.80k))
```

```
## # A tibble: 2,511 x 2
##    balance income
##      <dbl>  <dbl>
##  1       0      0
##  2       0   1000
##  3       0   2000
##  4       0   3000
##  5       0   4000
##  6       0   5000
##  7       0   6000
##  8       0   7000
##  9       0   8000
## 10       0   9000
## # ... with 2,501 more rows
```

```r
grid.tbl.7
```

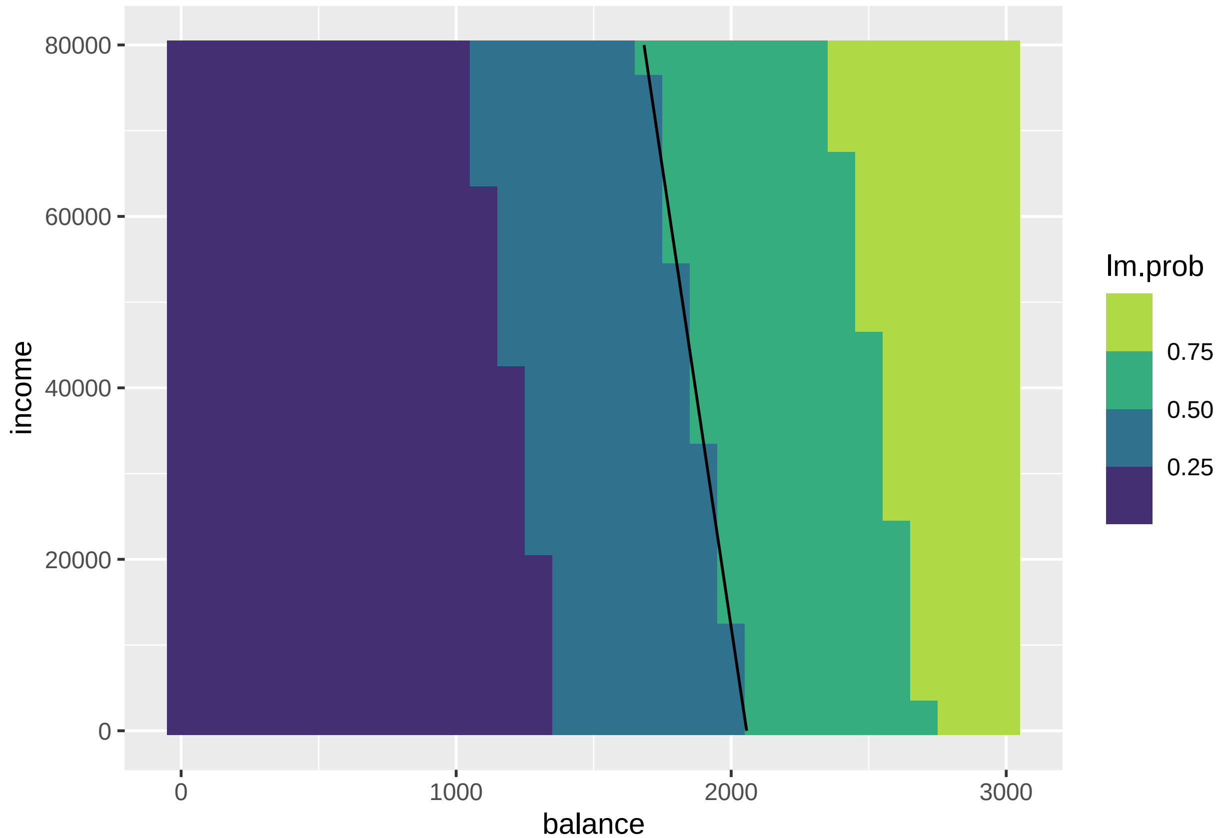
```
## # A tibble: 2,511 x 2
##    balance income
##      <dbl>  <dbl>
##  1       0      0
##  2       0   1000
##  3       0   2000
##  4       0   3000
##  5       0   4000
##  6       0   5000
##  7       0   6000
##  8       0   7000
##  9       0   8000
## 10       0   9000
## # ... with 2,501 more rows
```

```r
pred7 <- predict(linear.model2, grid.tbl.7)
maxVal <- max(pred7)
grid.tbl.7 <- grid.tbl.7 %>%
  mutate(lm.prob = pred7/maxVal) %>%
  mutate(lm.prob = ifelse(lm.prob < 0, 0, ifelse(lm.prob > 1, 1, lm.prob)))# %>%
  #mutate(p = ifelse(lm.prob < .5, "No", "Yes"))
grid.tbl.7
```

```
## # A tibble: 2,511 x 3
##    balance income lm.prob
##      <dbl>  <dbl>   <dbl>
##  1       0      0       0
##  2       0   1000       0
##  3       0   2000       0
##  4       0   3000       0
##  5       0   4000       0
##  6       0   5000       0
##  7       0   6000       0
##  8       0   7000       0
```

```
## 9        0   8000       0
## 10       0   9000       0
## # ... with 2,501 more rows
```

```
ggplot(grid.tbl.7, aes(balance, income, z=lm.prob,fill = lm.prob)) +
  geom_raster() +
  stat_contour(breaks=c(0.5), color="black")+
  scale_fill_viridis_b()
```



8. Does default change depending on whether somebody is a student or not? Illustrate your answer using a plot using facets.

```
trainerSYes <- trainer %>%
  dplyr::filter(student == "Yes")

trainerSNo <- trainer %>%
  dplyr::filter(student == "No")

linear.modelSYes <- lm(default_1~balance+income, data = trainerSYes)
linear.modelSNo <- lm(default_1~balance+income, data = trainerSNo)
```

```
grid.vec.3k = seq(0,3000, by=100)
grid.vec.80k = seq(0,80000, by=1000)
(grid.tbl.8 <- expand_grid(balance=grid.vec.3k, income=grid.vec.80k))
```

```
## # A tibble: 2,511 x 2
##    balance income
##      <dbl>  <dbl>
```

```
## 1       0      0
## 2       0   1000
## 3       0   2000
## 4       0   3000
## 5       0   4000
## 6       0   5000
## 7       0   6000
## 8       0   7000
## 9       0   8000
## 10      0   9000
## # ... with 2,501 more rows
```

```
pred8Yes <- predict(linear.modelSYes, grid.tbl.8)
pred8No <- predict(linear.modelSNo, grid.tbl.8)
maxValYes <- max(pred8Yes)
maxValNo <- max(pred8No)

grid.tbl.8.yes <- grid.tbl.8 %>%
  mutate(lm.prob.yes = pred8Yes/maxValYes) %>%
  mutate(lm.prob.yes = ifelse(lm.prob.yes < 0, 0, ifelse(lm.prob.yes > 1, 1, lm.prob.yes)))
grid.tbl.8.yes
```

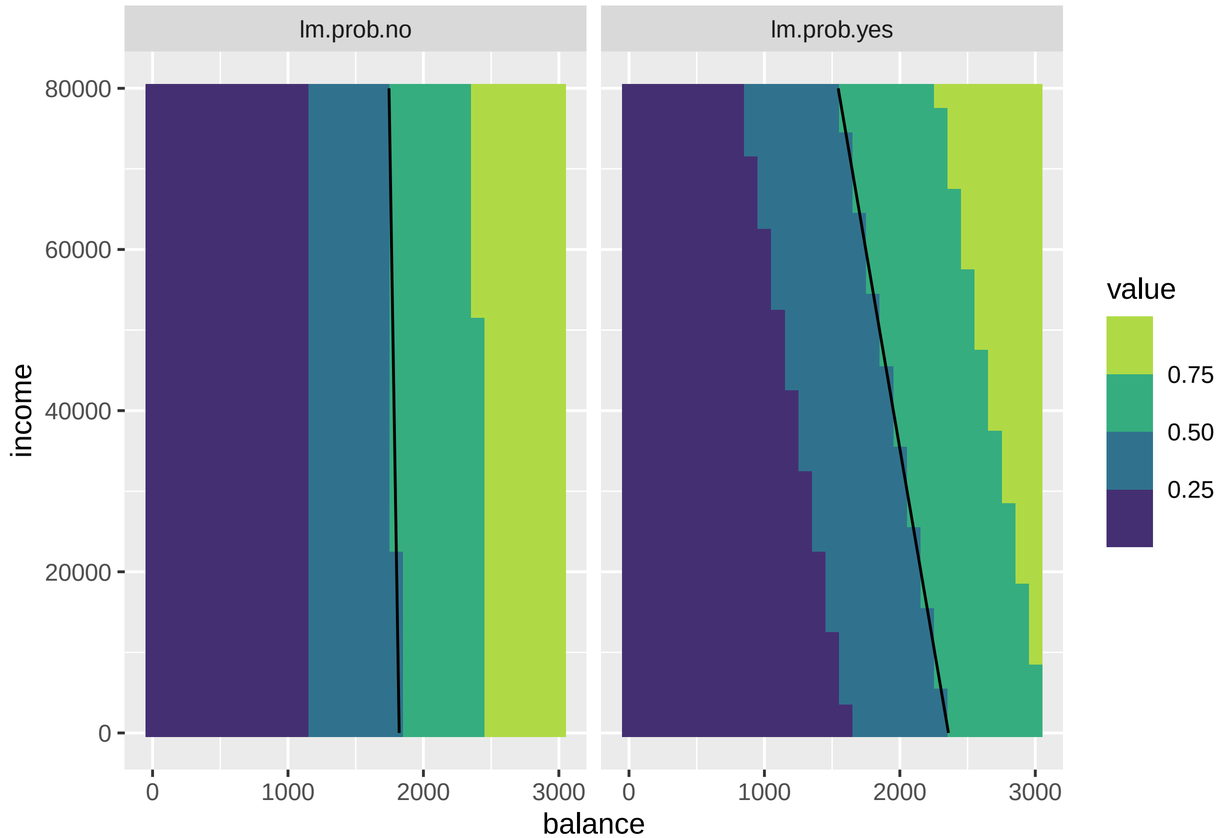
```
## # A tibble: 2,511 x 3
##    balance income lm.prob.yes
##      <dbl>  <dbl>       <dbl>
## 1        0      0           0
## 2        0   1000           0
## 3        0   2000           0
## 4        0   3000           0
## 5        0   4000           0
## 6        0   5000           0
## 7        0   6000           0
## 8        0   7000           0
## 9        0   8000           0
## 10       0   9000           0
## # ... with 2,501 more rows
```

```
grid.tbl.8.no <- grid.tbl.8 %>%
  mutate(lm.prob.no = pred8No/maxValNo) %>%
  mutate(lm.prob.no = ifelse(lm.prob.no < 0, 0, ifelse(lm.prob.no > 1, 1, lm.prob.no)))
grid.tbl.8.no
```

```
## # A tibble: 2,511 x 3
##    balance income lm.prob.no
##      <dbl>  <dbl>      <dbl>
## 1        0      0          0
## 2        0   1000          0
## 3        0   2000          0
## 4        0   3000          0
## 5        0   4000          0
## 6        0   5000          0
## 7        0   6000          0
## 8        0   7000          0
## 9        0   8000          0
## 10       0   9000          0
```

```
## # ... with 2,501 more rows
```

```r
full_join(grid.tbl.8.yes, grid.tbl.8.no, by = c("balance", "income")) %>%
  pivot_longer(3:4) %>%
  ggplot(aes(balance, income, z=value,fill = value)) +
  geom_raster() +
  stat_contour(breaks=c(0.5), color="black")+
  scale_fill_viridis_b() +
  facet_wrap(.~name)
```



lm.prob.yes represents students, and lm.prob.no represents non-students. The probability of defaulting is, I think, lower for students because more of the graph is less than p = 0.5. The probability of defaulting is lower for students given the same balance.

9. Create a linear model that uses `student`, `balance`, and `income` to predict `default`. What is the misclassification rate of this model? Are the results better than the model created in 6?

trainer, tester

```r
linear.model3 <- lm(default_1~balance+income+student, data = trainer)
linear.model3
```

```
##
## Call:
## lm(formula = default_1 ~ balance + income + student, data = trainer)
##
## Coefficients:
## (Intercept)      balance       income   studentYes
##  -7.876e-02    1.297e-04    2.171e-07   -1.508e-02
```

```r
pred4 <- predict(linear.model3, tester)
value <- max(pred4)
tester3 <- tester %>%
  mutate(lm = pred4/value) %>%
  mutate(lm = ifelse(lm < 0, 0, ifelse(lm > 1, 1, lm))) %>%# added ifelse(lm.prob.no > 1, 1, lm.prob.no
  mutate(p = ifelse(lm < .5, "No", "Yes"))
#max "maximum model value" is pred3? 0.2379911
tester3
```

```
## # A tibble: 2,000 x 6
##    default student balance income     lm p
##    <fct>   <fct>     <dbl>  <dbl>  <dbl> <chr>
##  1 No      Yes        920.  7492. 0.112  No
##  2 No      Yes          0  21871. 0      No
##  3 No      Yes       1221. 13269. 0.278  No
##  4 No      No         237. 28252. 0      No
##  5 No      No         286. 45042. 0      No
##  6 No      No           0  50265. 0      No
##  7 No      Yes        528. 17637. 0      No
##  8 No      No        1095. 26465. 0.284  No
##  9 No      No         643. 41474. 0.0562 No
## 10 No      No         495. 54385. 0      No
## # ... with 1,990 more rows
```

misclassification rate:

```r
mean(tester3$default != tester3$p)
```

```
## [1] 0.0605
```

1] 0.036 for #6 [1] 0.0605 for this one.

The misclassification rate is higher for this model. (why? I'd think more explanatory variables = more accuracy. I guess not.)

10. Plot the probability and the decision boundary for the model created in 9.

to plot this, I must use a grid similar to #8. (to find misclassification rate, I predict defaulting using the original table and compare the two columns)

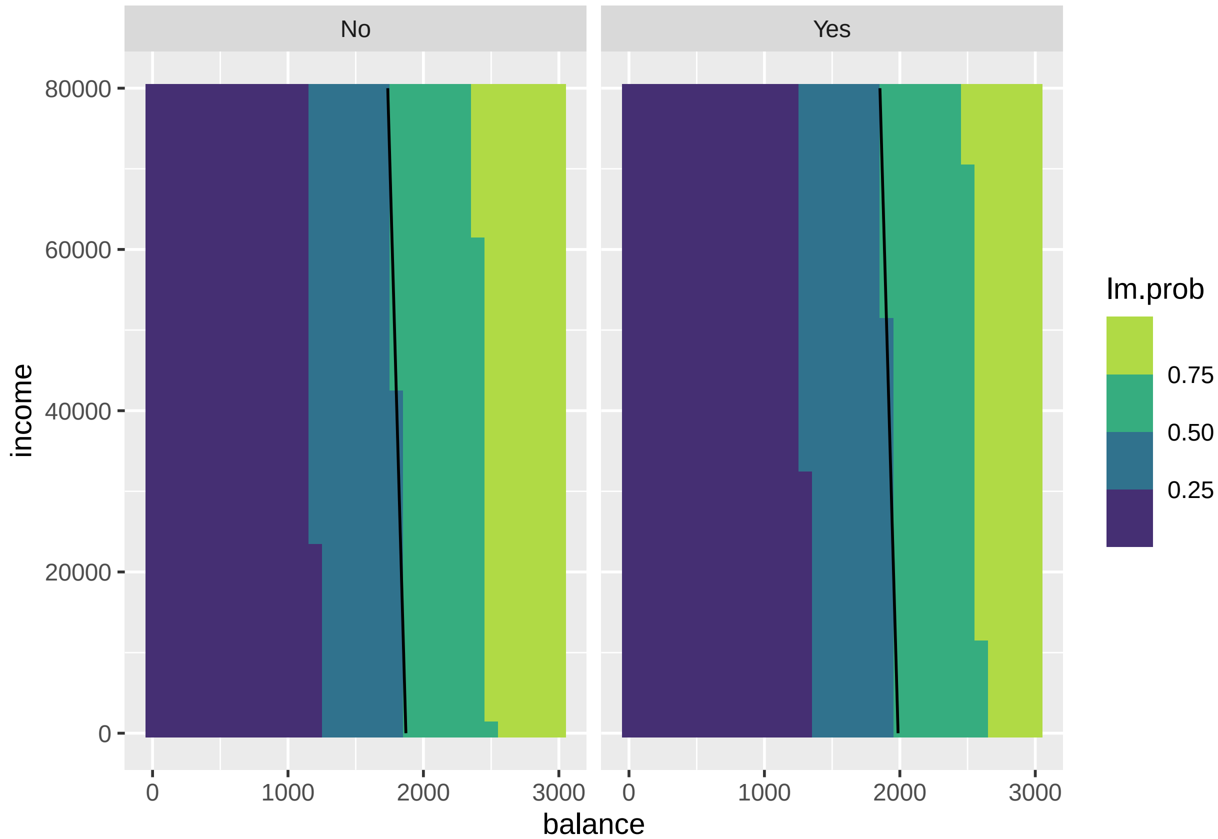
```r
tester3
```

```
## # A tibble: 2,000 x 6
##    default student balance income     lm p
##    <fct>   <fct>     <dbl>  <dbl>  <dbl> <chr>
##  1 No      Yes        920.  7492. 0.112  No
##  2 No      Yes          0  21871. 0      No
##  3 No      Yes       1221. 13269. 0.278  No
##  4 No      No         237. 28252. 0      No
##  5 No      No         286. 45042. 0      No
##  6 No      No           0  50265. 0      No
##  7 No      Yes        528. 17637. 0      No
##  8 No      No        1095. 26465. 0.284  No
##  9 No      No         643. 41474. 0.0562 No
## 10 No      No         495. 54385. 0      No
## # ... with 1,990 more rows
```

```
grid.vec.3k = seq(0,3000, by=100)
grid.vec.80k = seq(0,80000, by=1000)
grid.vec.student = seq(0,1, by = 1)

(grid.tbl.8 <- expand_grid(balance=grid.vec.3k, income=grid.vec.80k, student = grid.vec.student))
```

```
## # A tibble: 5,022 x 3
##    balance income student
##      <dbl>  <dbl>   <dbl>
##  1       0      0       0
##  2       0      0       1
##  3       0   1000       0
##  4       0   1000       1
##  5       0   2000       0
##  6       0   2000       1
##  7       0   3000       0
##  8       0   3000       1
##  9       0   4000       0
## 10       0   4000       1
## # ... with 5,012 more rows
```

```
grid.tbl.8 <- grid.tbl.8 %>%
  mutate(student = ifelse(student == 1, "Yes", "No")) %>%
  mutate(student = as.factor(student))

somePredVals <- predict(linear.model3, grid.tbl.8)
#somePredVals

maxSomePredVals <- max(somePredVals)
#maxSomePredVals

plot.grid.tbl.8 <- grid.tbl.8 %>%
  mutate(lm.prob = somePredVals/maxSomePredVals) %>%
  mutate(lm.prob = ifelse(lm.prob < 0, 0, ifelse(lm.prob > 1, 1, lm.prob)))#this line is not really nec

#plot.grid.tbl.8
# balance income student lm.prob

plot.grid.tbl.8 %>%
  ggplot(aes(balance, income, z = lm.prob, fill = lm.prob)) +
    geom_raster() +
    stat_contour(breaks = c(0.5), color = "black") +
    scale_fill_viridis_b() +
    facet_wrap(~student)
```



the two plots look extremely similar.