Classification-2

Andrew Noecker

2/23/2022

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)
```

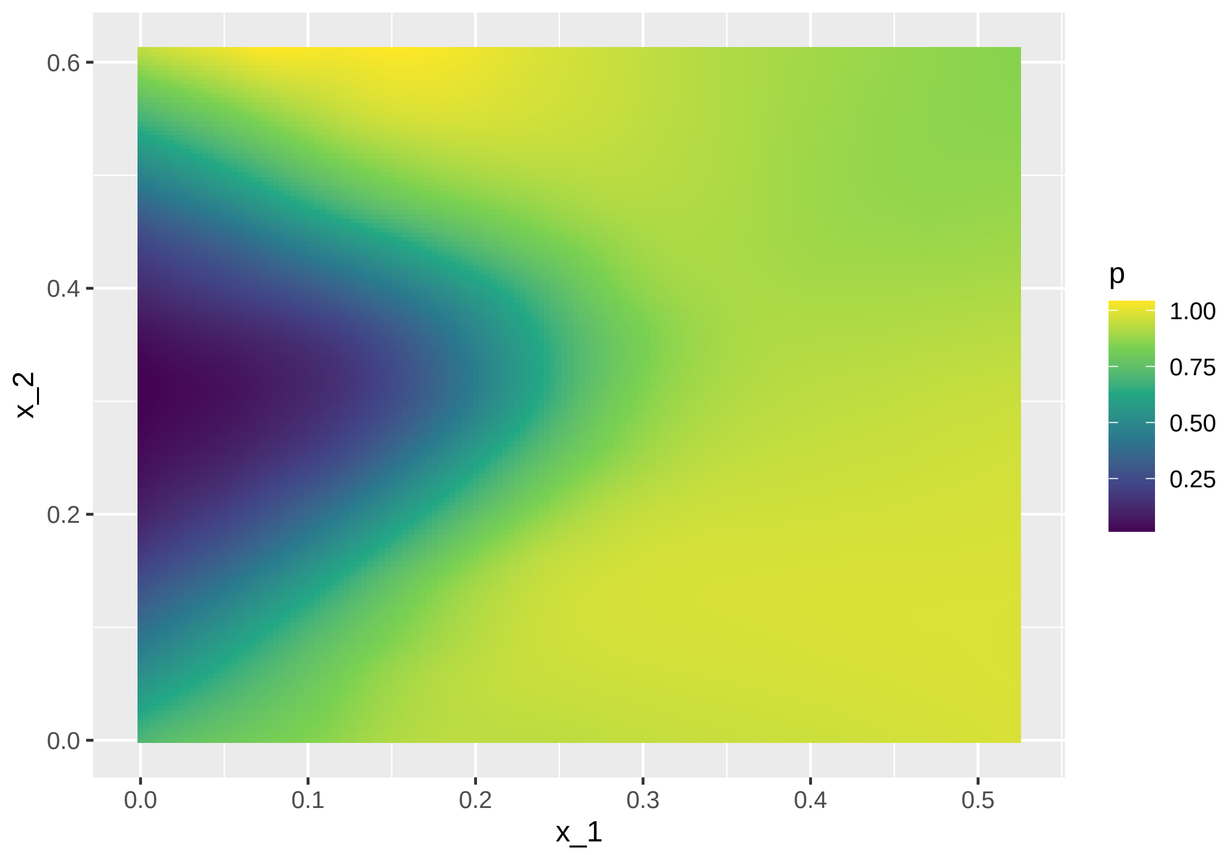
```
## [1] 200   3
```

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:

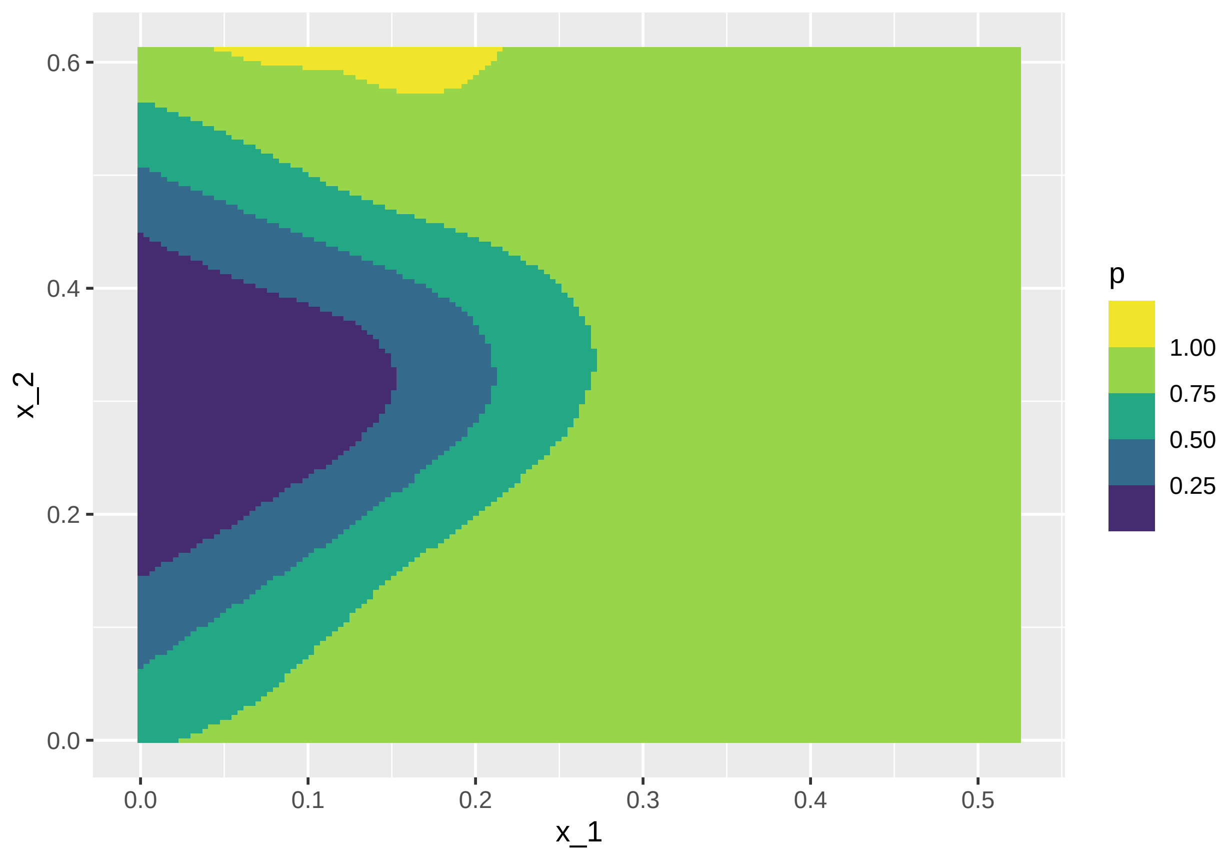
```
mnist.true.tbl <-  tibble(mnist_27$true_p)
```

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()
```

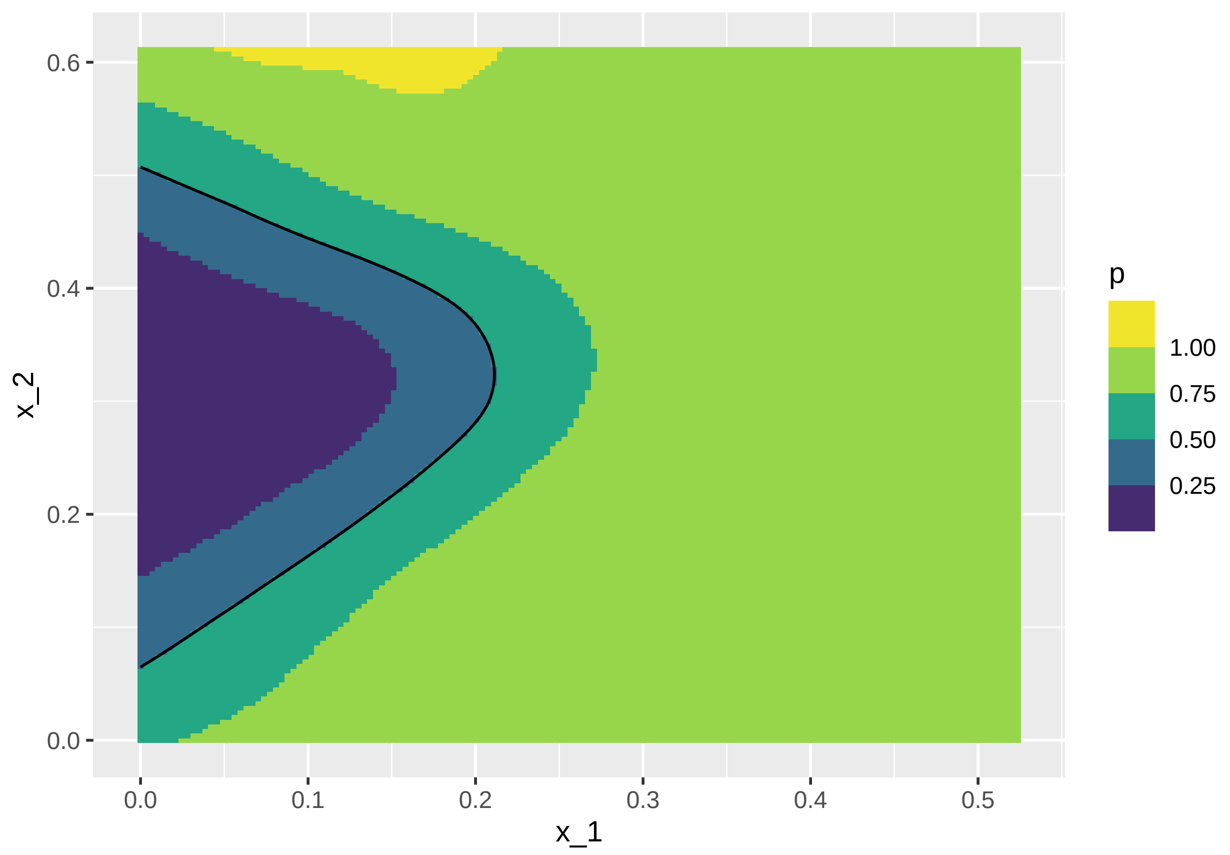


```r
ggplot(mnist.true.tbl, aes(x_1, x_2, fill = p)) +
  geom_raster() +
  scale_fill_viridis_b()
```



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:



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`.

```
grid.vec <- seq(from = 0, to = 1, by = 0.1)
```

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

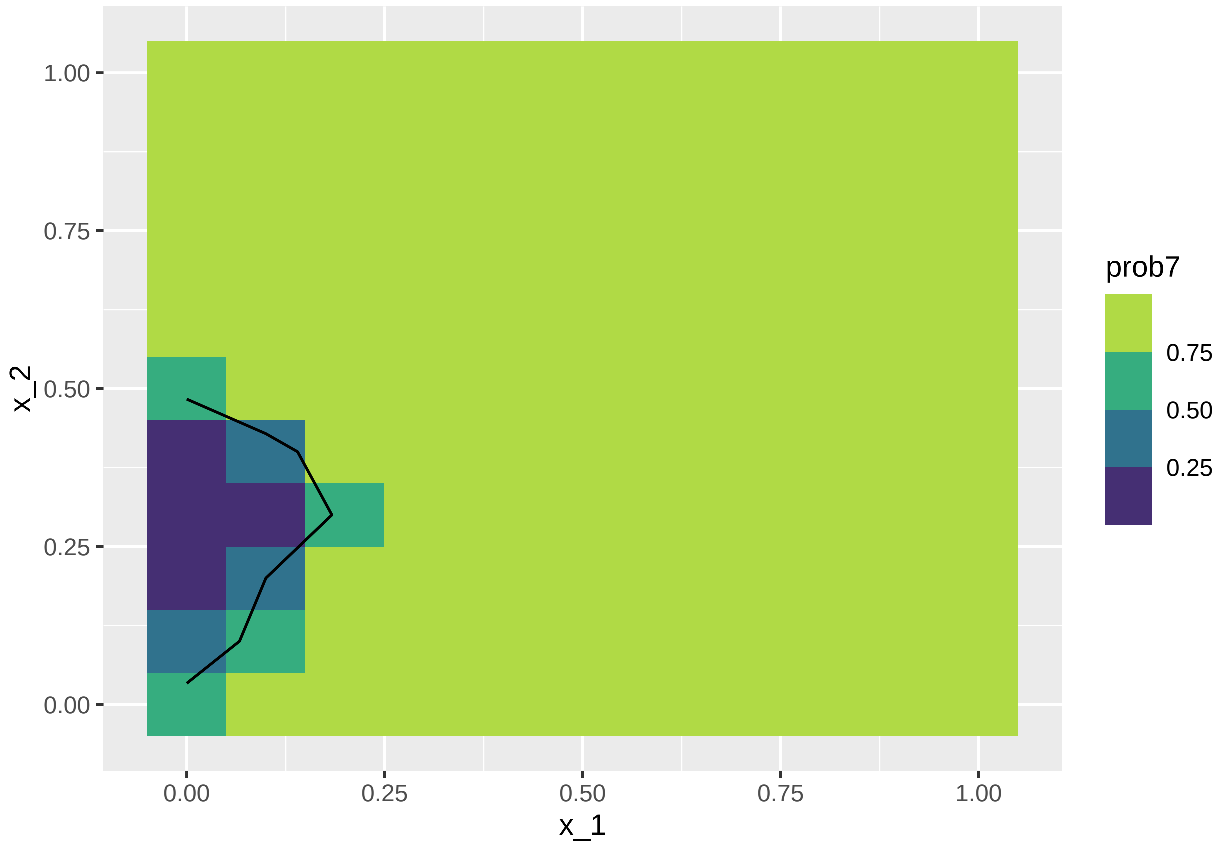
```
grid.tbl <- expand_grid(x_1 = grid.vec, x_2 = grid.vec)
```

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.

```
prob <- predict(knn.model, newdata = grid.tbl)
grid.tbl <- grid.tbl%>%
  mutate(prob2 = prob[,1],
         prob7 = prob[,2])
```

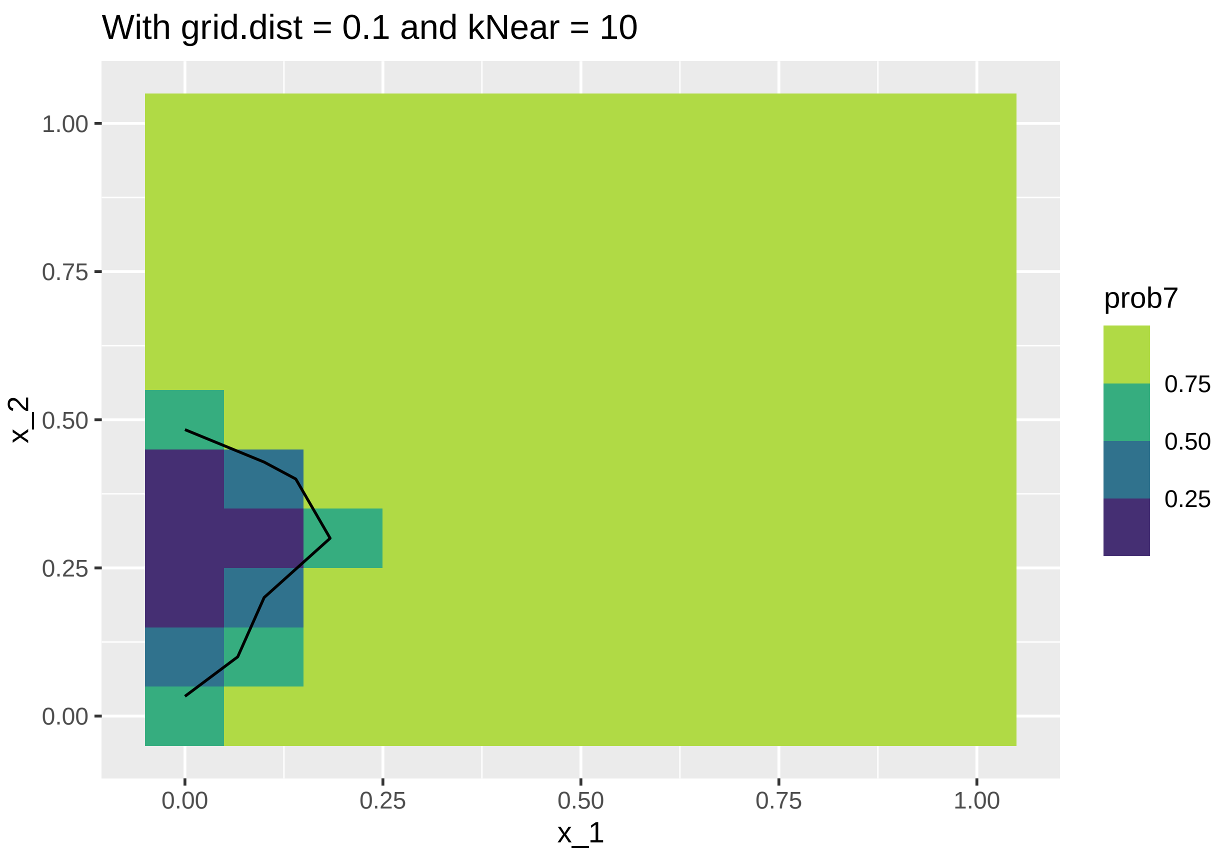
d. Use `grid.tbl` to plot the predicted probability across the unit grid and plot the *decision boundary*.

```
ggplot(grid.tbl, aes(x_1, x_2, z=prob7,fill = prob7)) +
  geom_raster() +
  stat_contour(breaks=c(0.5), color="black")+
  scale_fill_viridis_b()
```

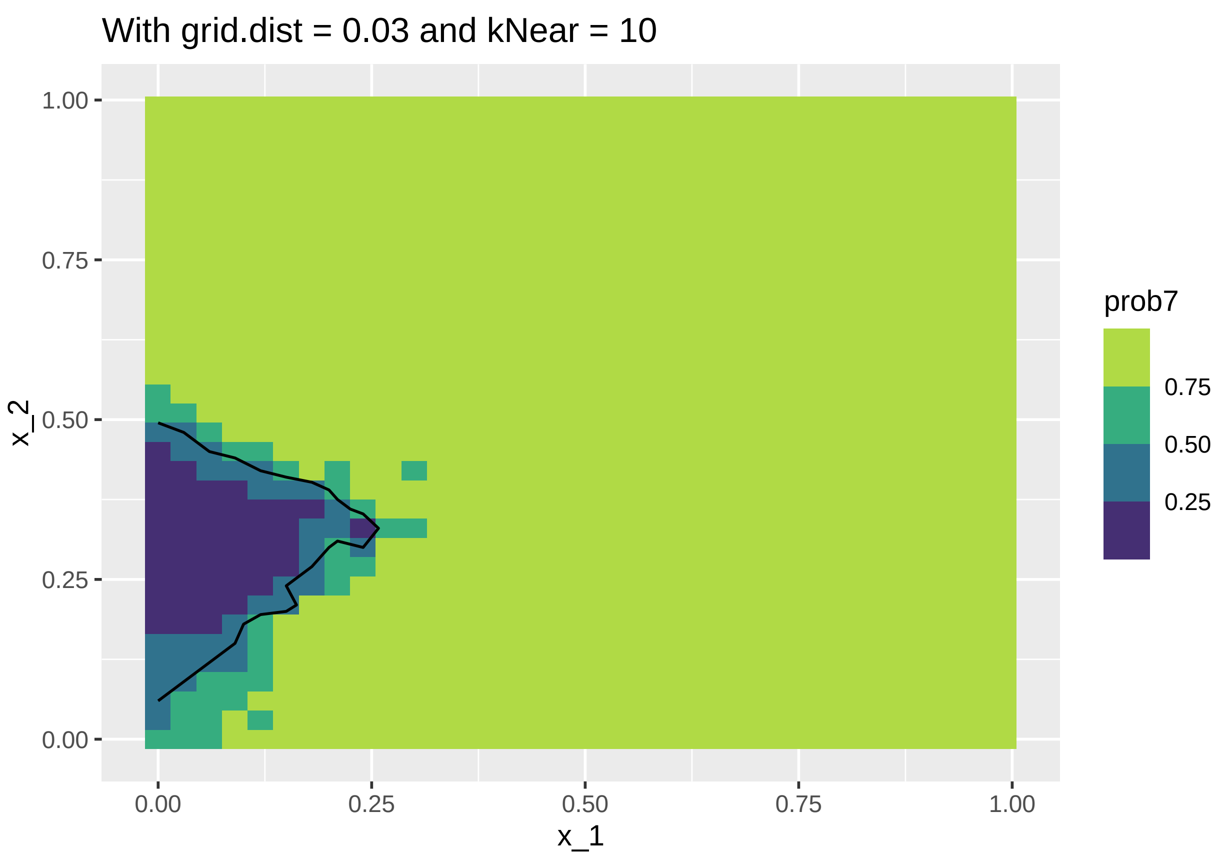


e. It seems your graph is too pixelated. Create a function `plot_knn_model(kNear, grid.dist, train.tbl)` that trains a KNN model with parameter `kNear` on `train.tbl` and displays the value of the probability of being a 7 on a grid of points generated every `grid.dist`. Evaluate your function using `grid.dist`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

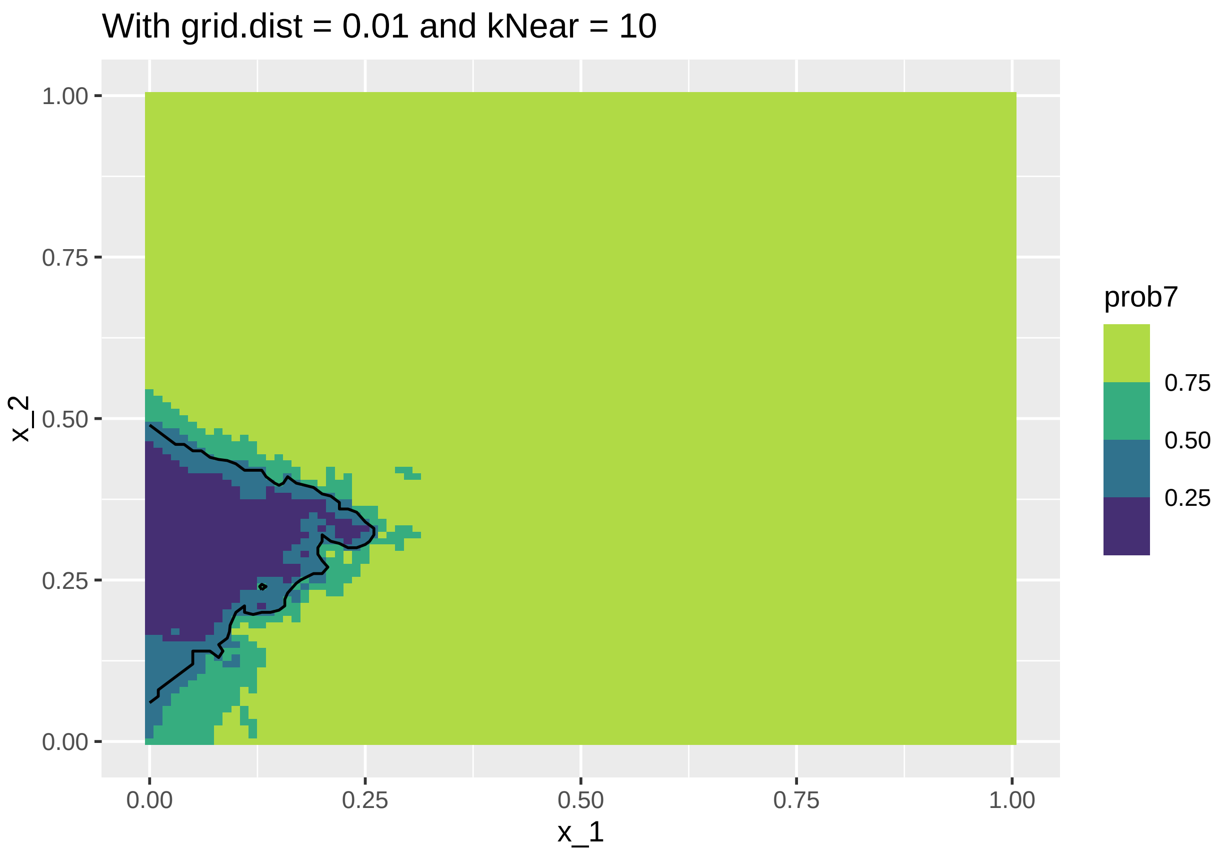
```
plot_knn_model <- function(kNear, grid.dist, train.tbl){

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

  grid.vec <- seq(from = 0, to = 1, by = grid.dist)
  grid.tbl <- expand_grid(x_1 = grid.vec, x_2 = grid.vec)

  prob <- predict(knn.model, newdata = grid.tbl)
  grid.tbl <- grid.tbl%>%
  mutate(prob2 = prob[,1],
         prob7 = prob[,2])

  ggplot(grid.tbl, aes(x_1, x_2, z=prob7,fill = prob7)) +
    geom_raster() +
    stat_contour(breaks=c(0.5), color="black")+
    scale_fill_viridis_b()+
    labs(title = str_c("With grid.dist = ", grid.dist, " and kNear = ", kNear))
}


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



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

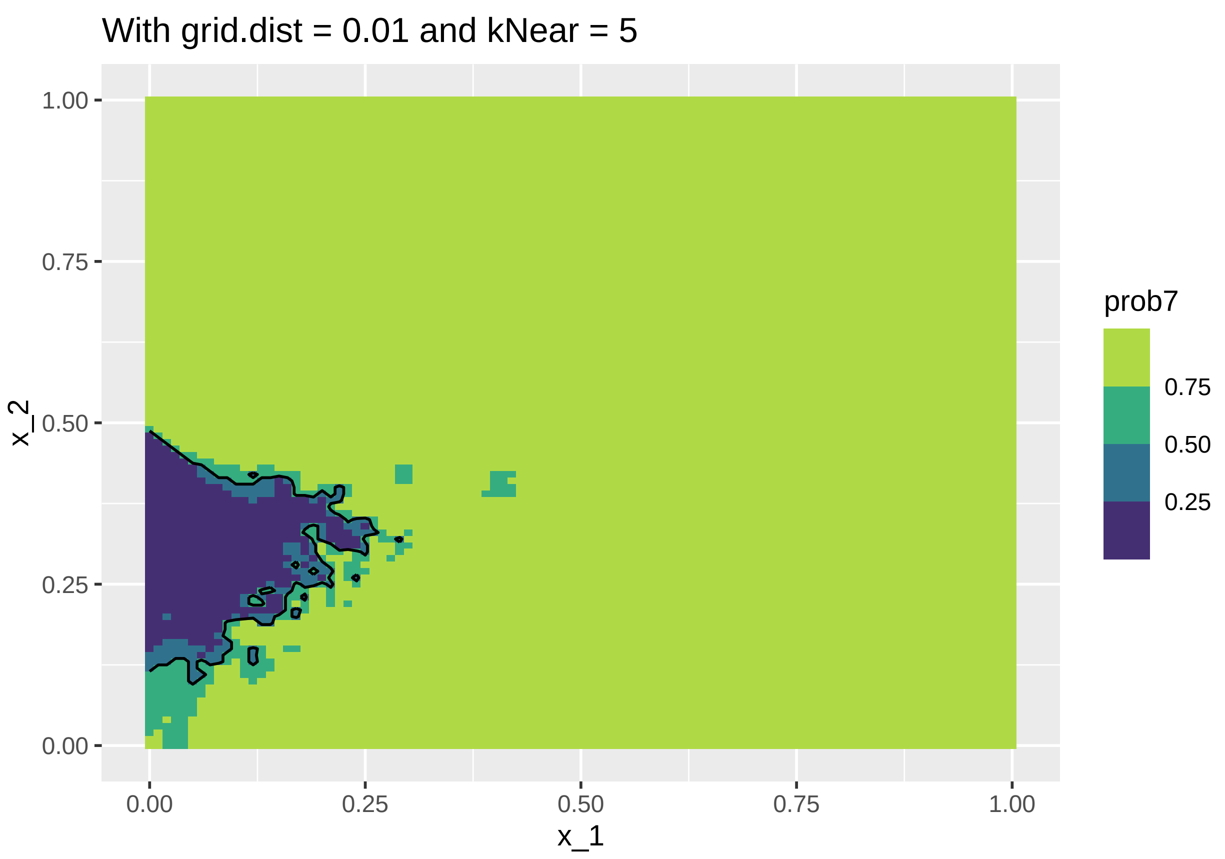


```
plot_knn_model(10, 0.01, mnist.train.tbl)
```

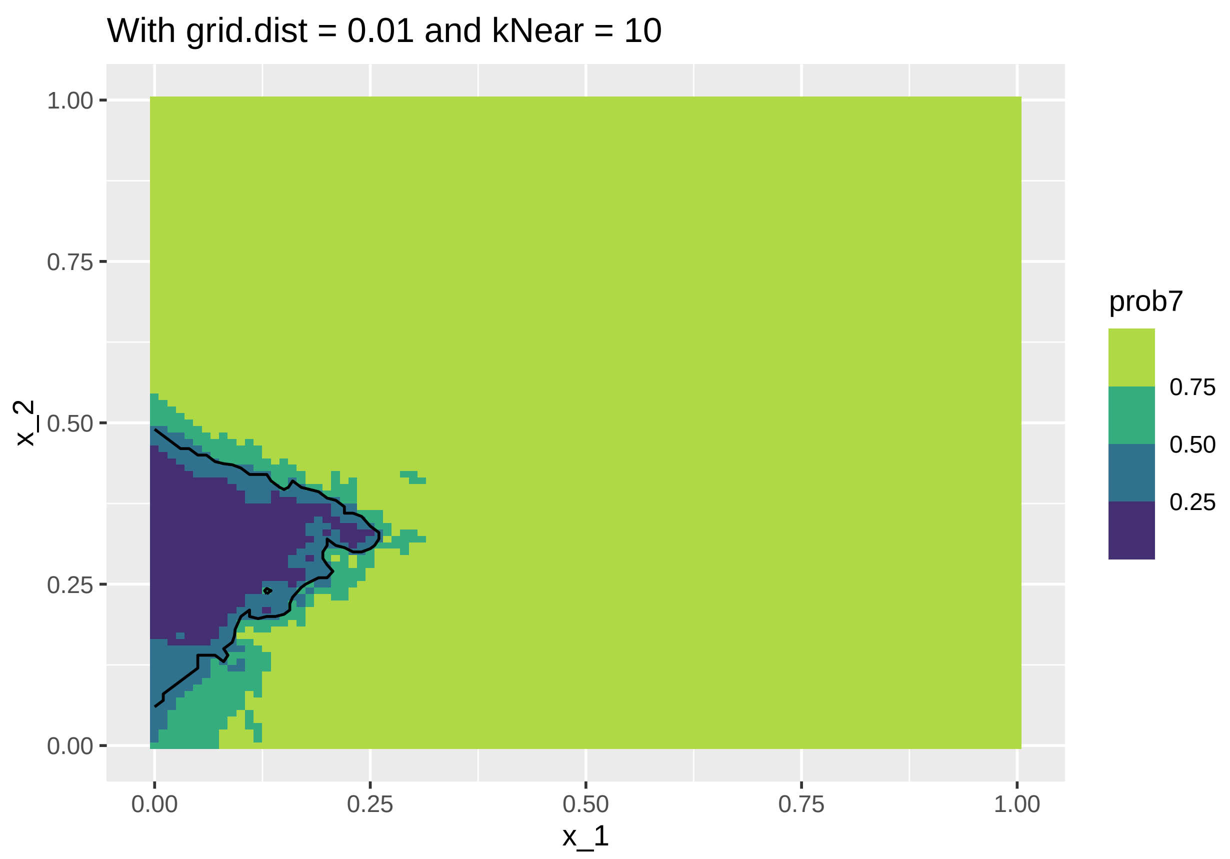


3. Experiment plotting with different values of k (say from k=5 to k=50, using steps of 5). Which decision boundary looks more similar to the Bayes boundary? Is this consistent with the optimal value of `k` that you found using in point 4 of our last activity, `3_Classification.Rmd`?

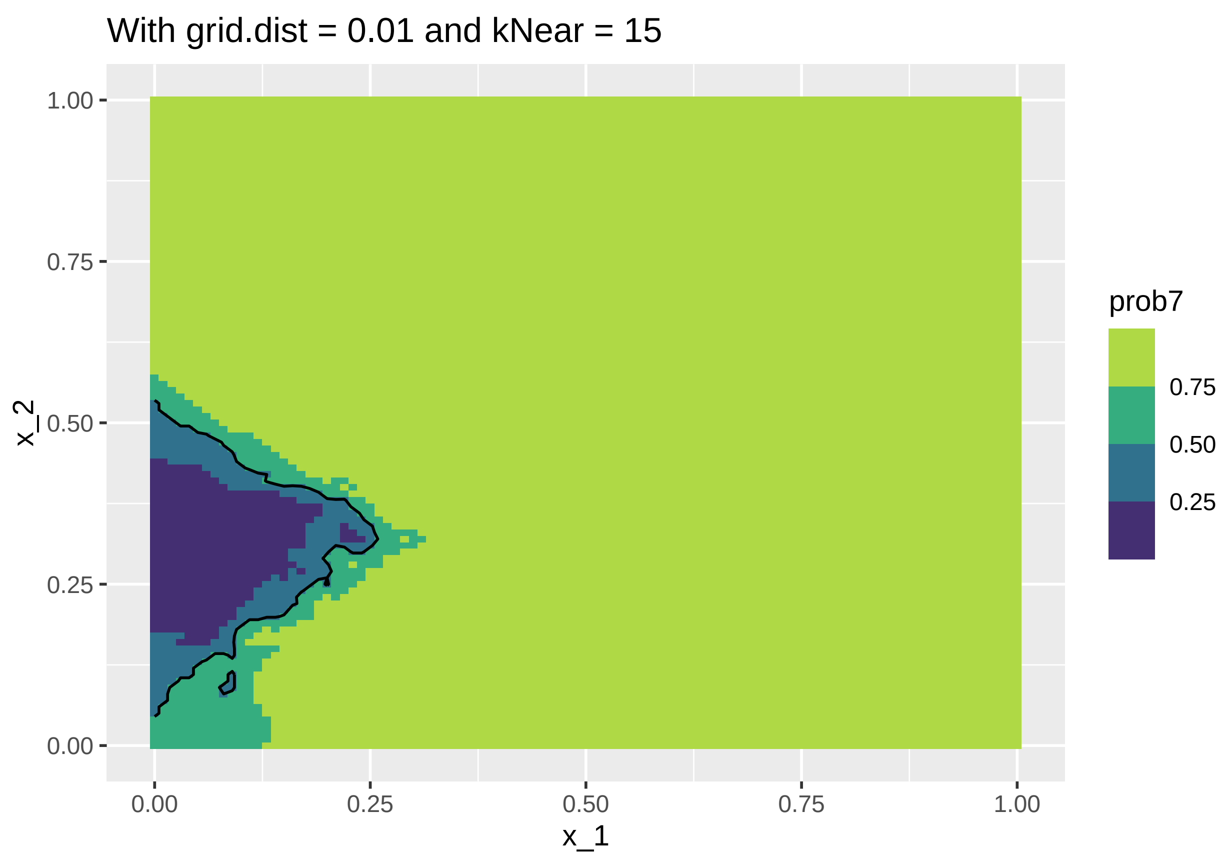
```
plot_knn_model(5, 0.01, mnist.train.tbl)
```



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

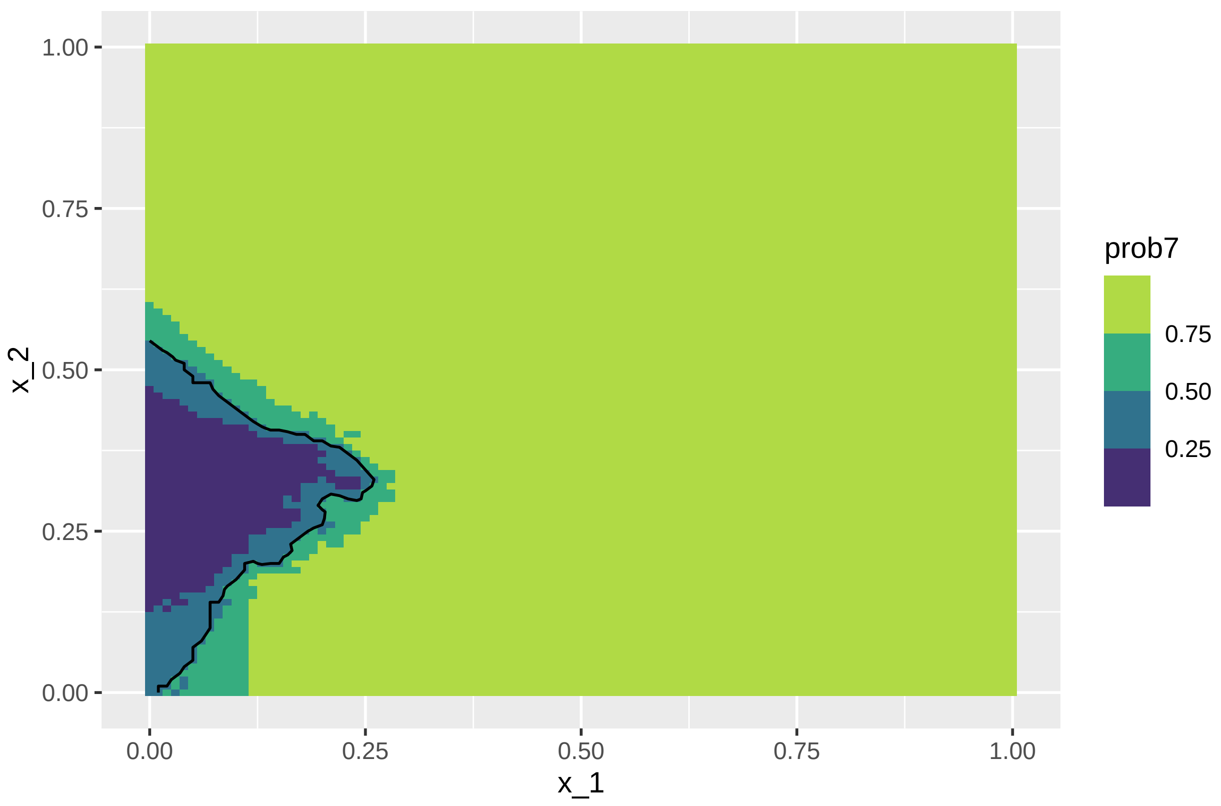


```
plot_knn_model(15, 0.01, mnist.train.tbl)
```

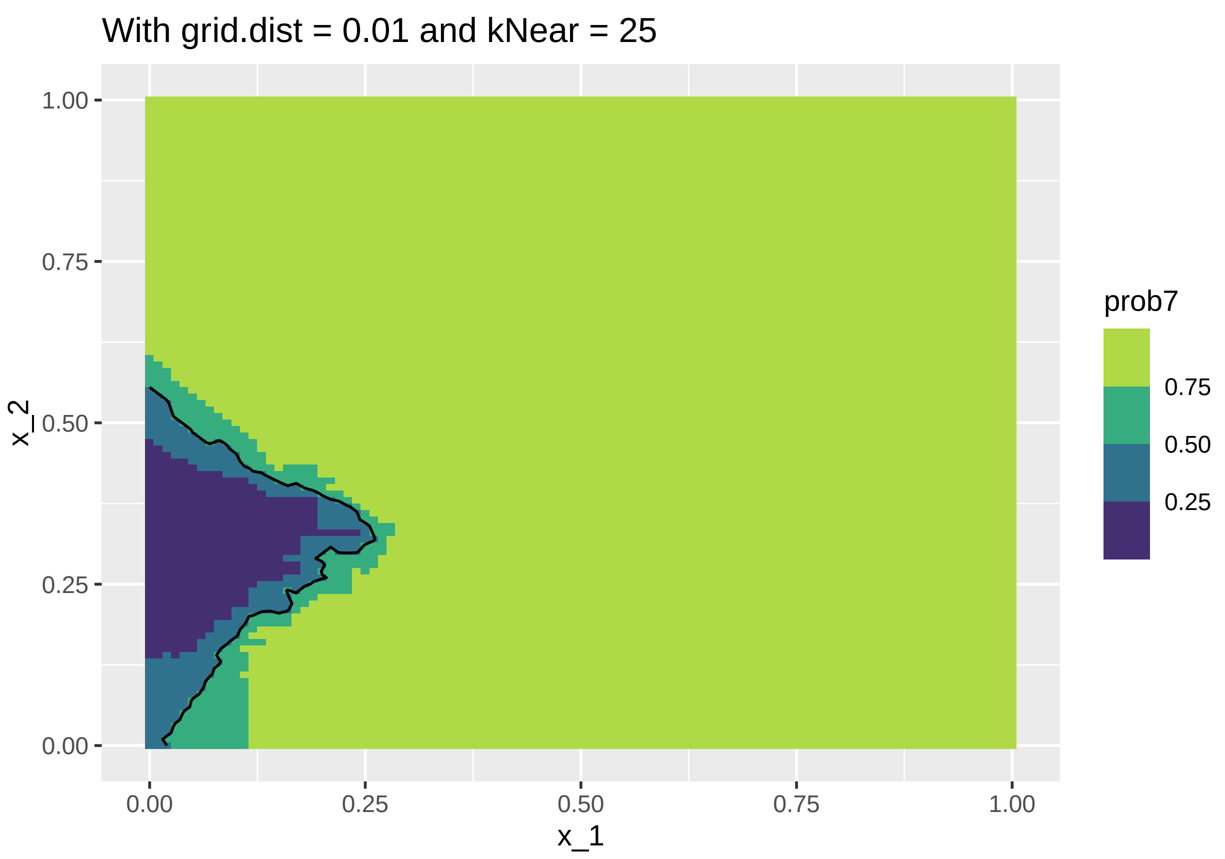


```r
plot_knn_model(20, 0.01, mnist.train.tbl)
```

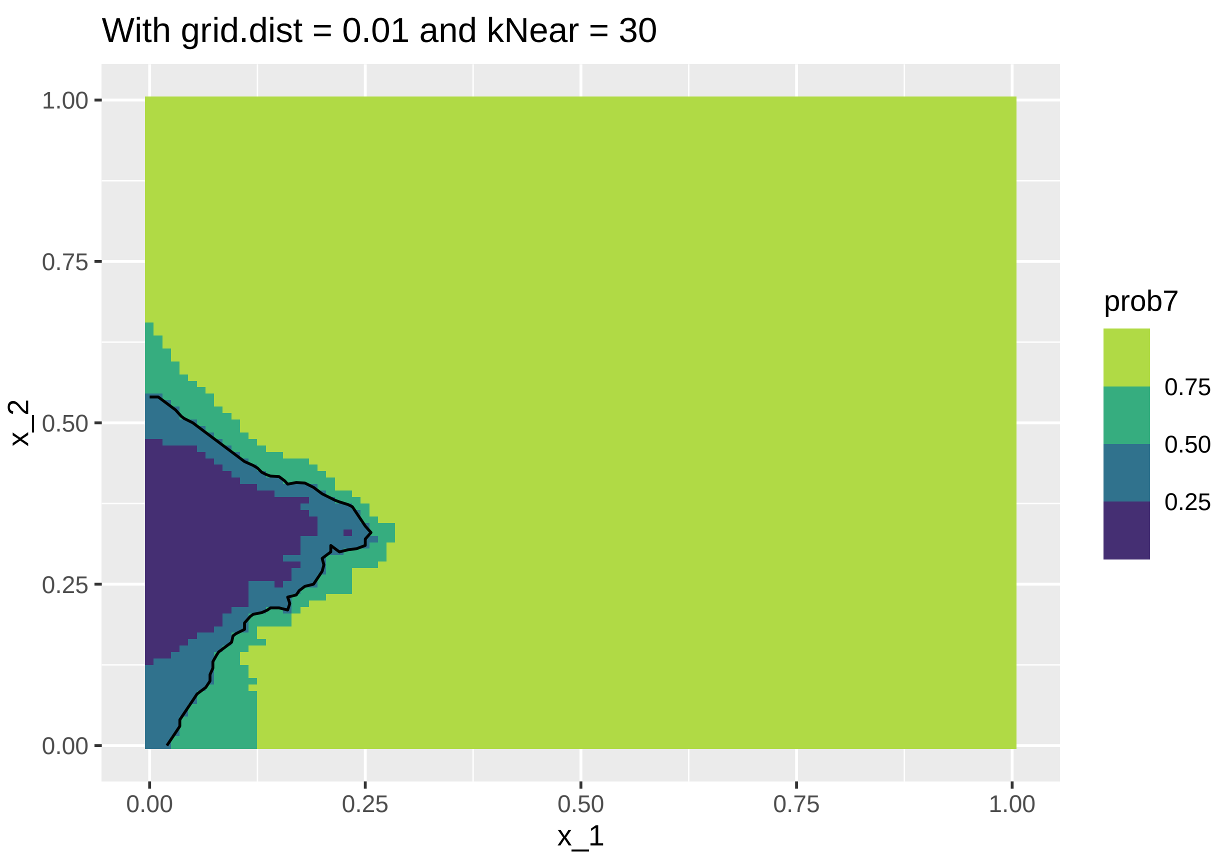
With grid.dist = 0.01 and kNear = 20



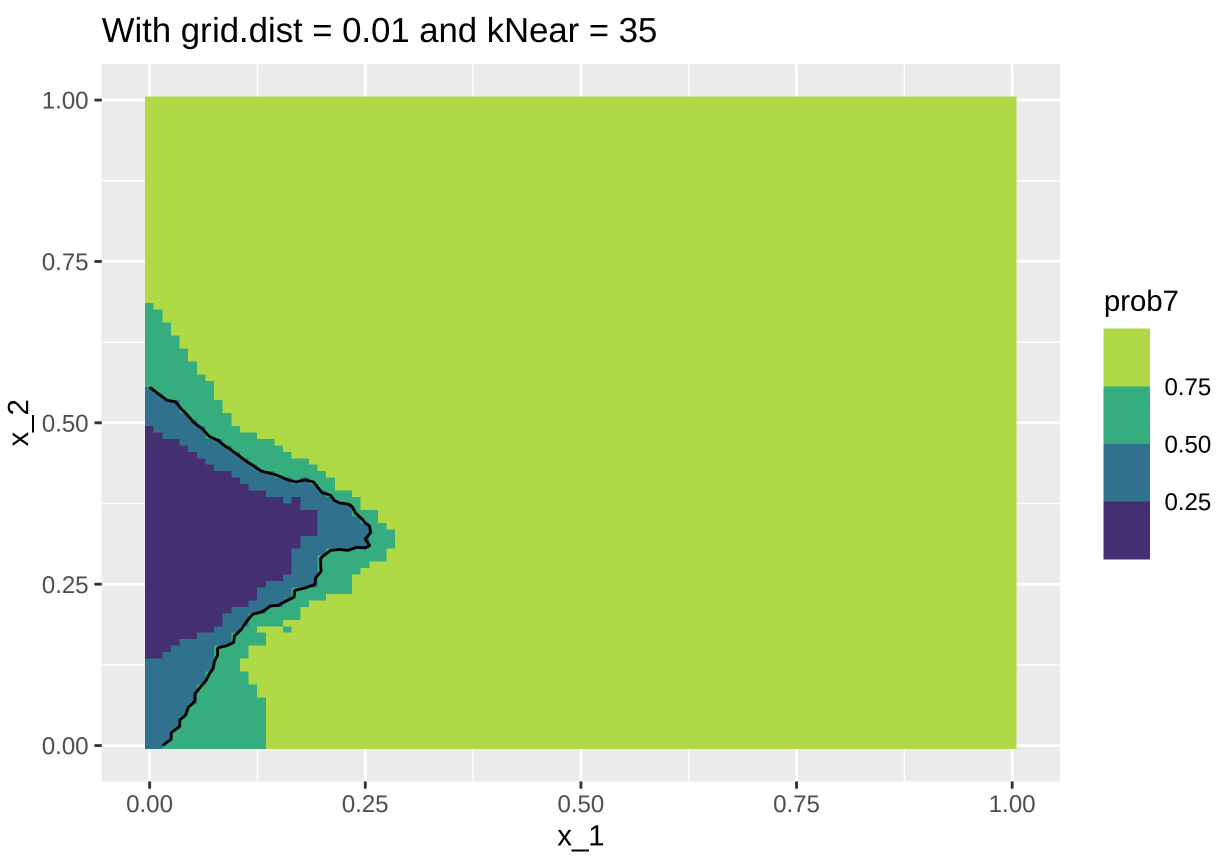
```
plot_knn_model(25, 0.01, mnist.train.tbl)
```



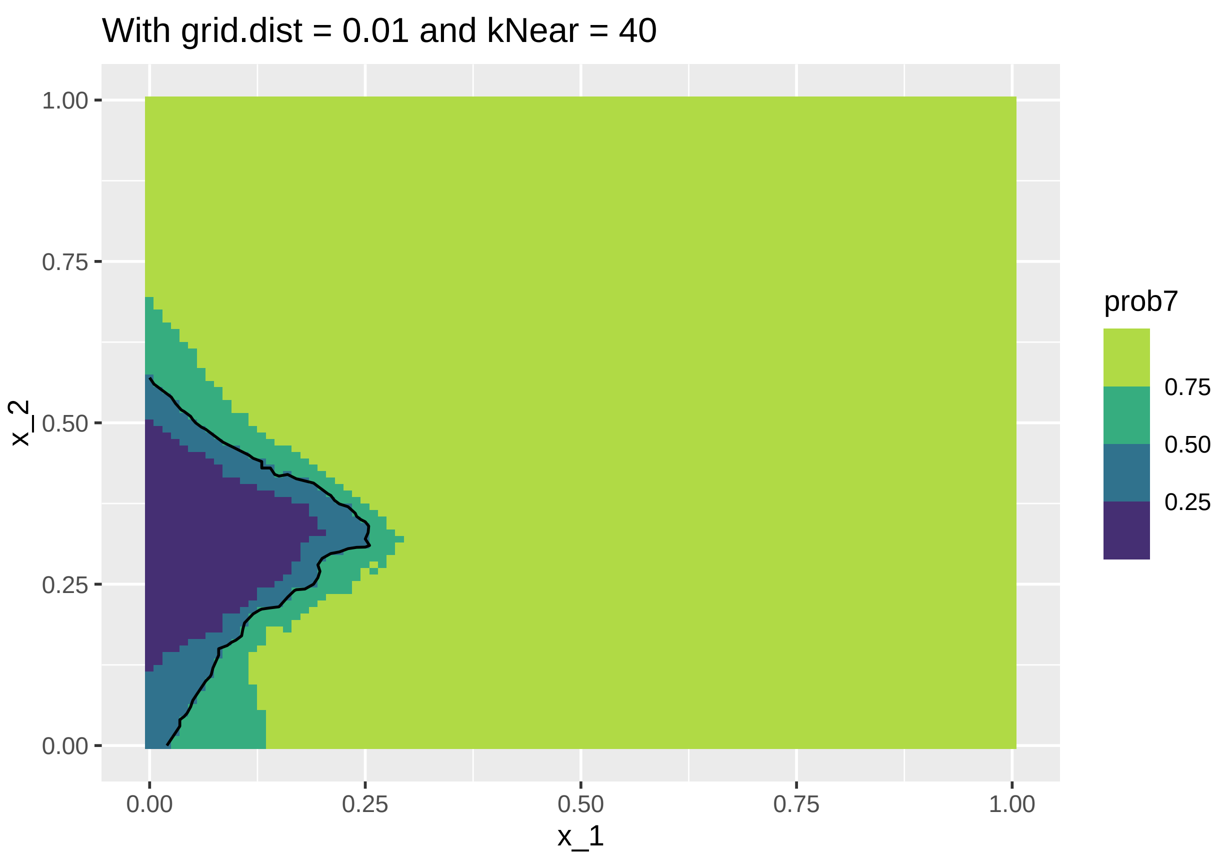
```
plot_knn_model(30, 0.01, mnist.train.tbl)
```



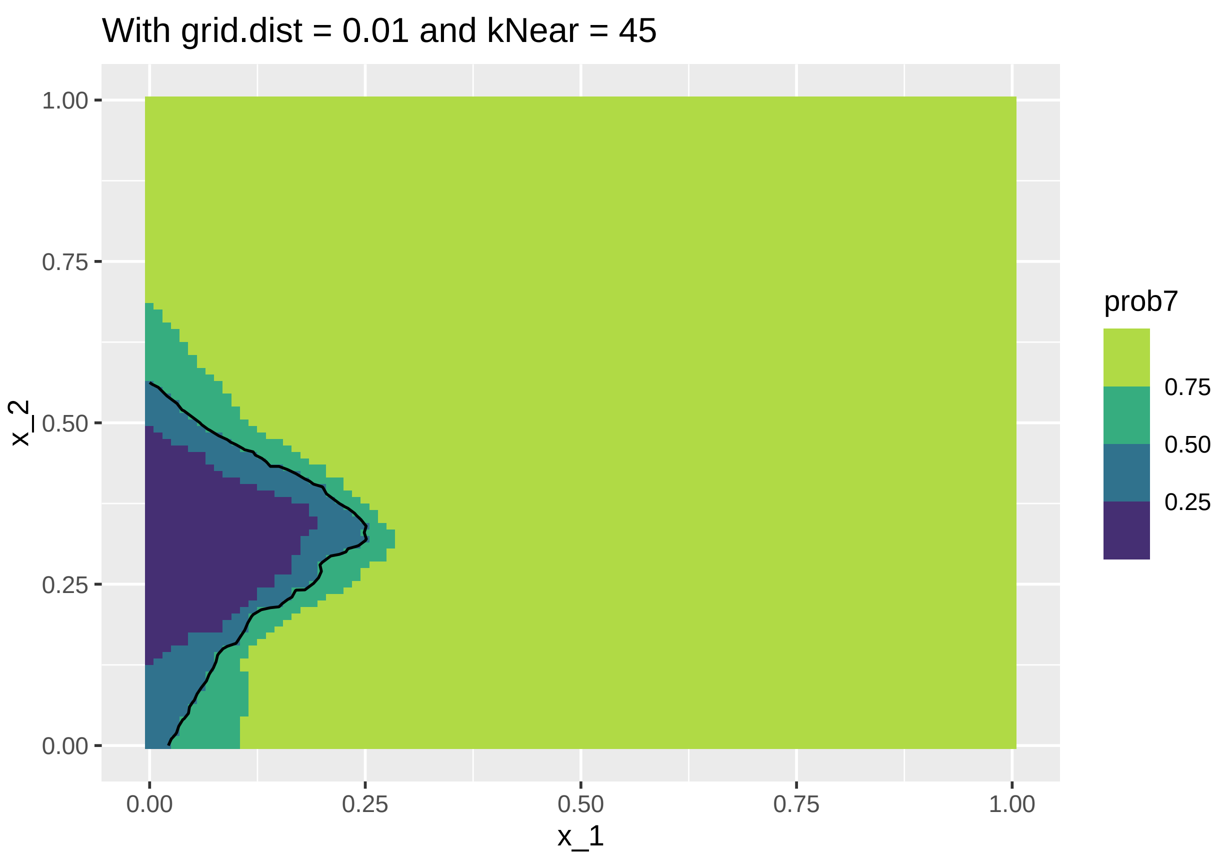
```
plot_knn_model(35, 0.01, mnist.train.tbl)
```



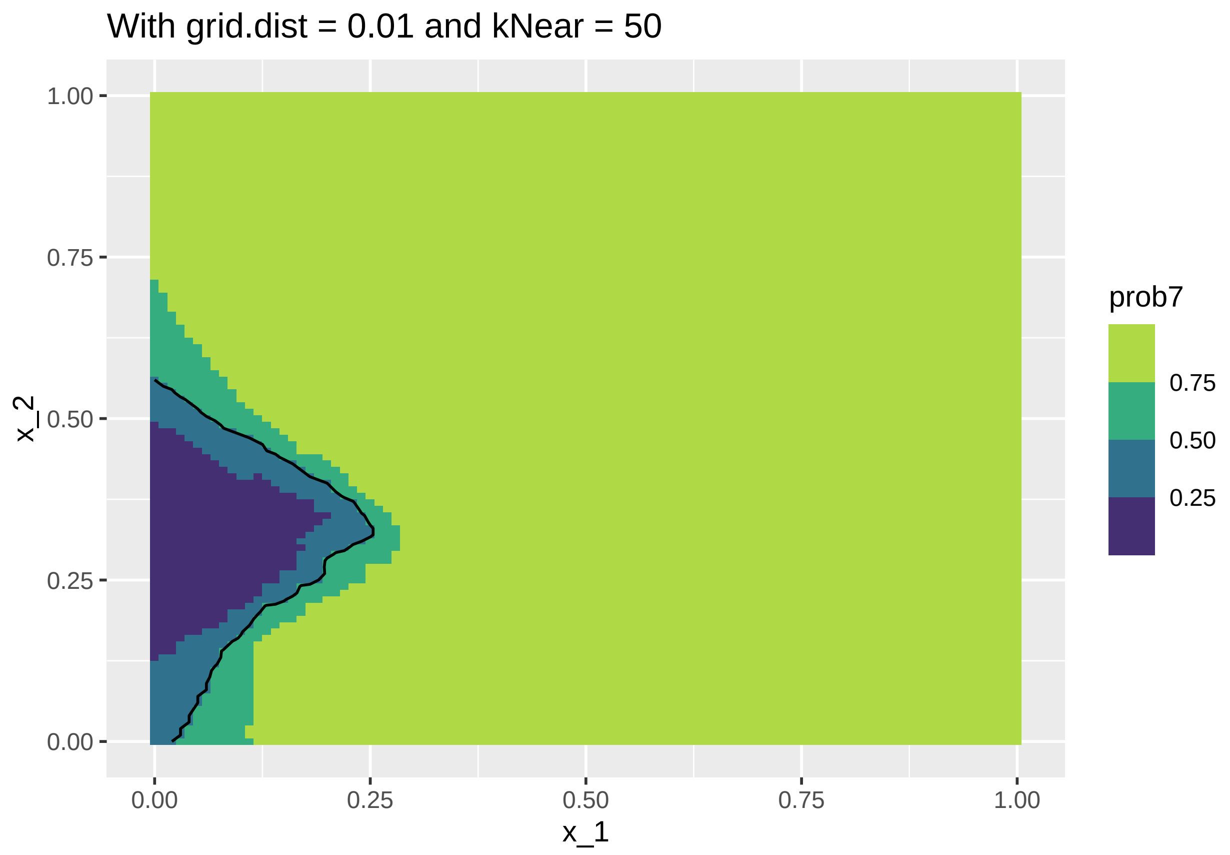
```r
plot_knn_model(40, 0.01, mnist.train.tbl)
```



```
plot_knn_model(45, 0.01, mnist.train.tbl)
```



```
plot_knn_model(50, 0.01, mnist.train.tbl)
```

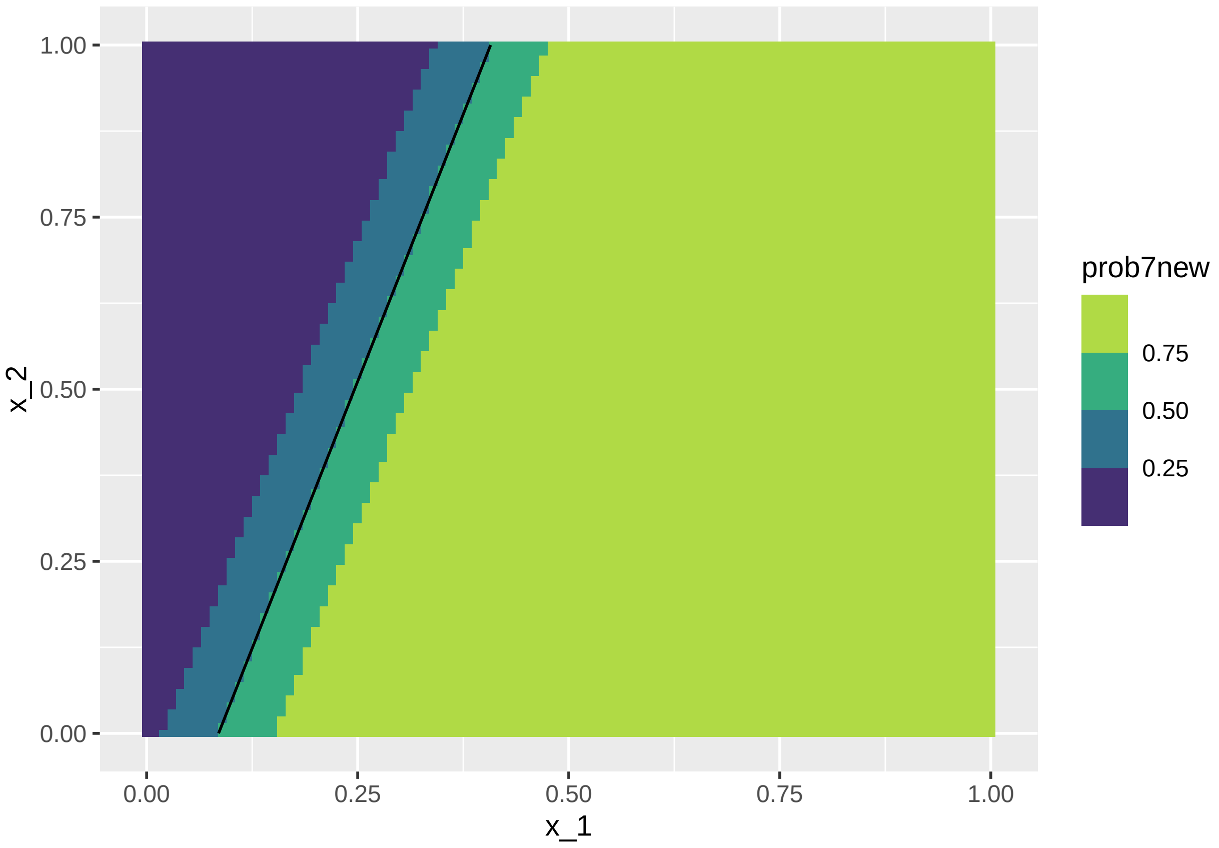


I think that having the higher kNear leads to the more similar Bayes boundary. With kNear = 50 we have a decision boundary that looks pretty similar to what we saw in the preface of this section, but kNear = 40 yields a pretty similar one as well. In the prior document, we found the optimal kNear to be 41, which is pretty close with what we found here, so our results mostly line up.

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 2s are encoded as zeros and 7s 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?

```
mnist.train.tbl <- mnist.train.tbl%>%
  mutate(is7 = ifelse(y == "7", 1, 0))

mnist.lm <- lm(is7 ~ x_1+x_2, data = mnist.train.tbl)

new.grid.vec <- seq(0, 1, 0.01)
new.grid.tbl <- expand_grid(x_1 = new.grid.vec, x_2 = new.grid.vec)

prob <- predict(mnist.lm, new.grid.tbl)

new.grid.tbl2 <- new.grid.tbl %>%
  mutate(prob7new = prob)%>%
  mutate(prob7new = ifelse(prob7new < 0, 0, prob7new),
         prob7new = ifelse(prob7new > 1, 1, prob7new))

ggplot(new.grid.tbl2, aes(x_1, x_2, z=prob7new, fill = prob7new)) +
    geom_raster() +
    stat_contour(breaks=c(0.5), color="black")+
    scale_fill_viridis_b()
```



This boundary is (obviously) a straight line, but it's much simpler/dry cut than the boundary we generated with the KNN model. While the KNN model boundary may overfit at times, this one appears to be somewhat simplistic.

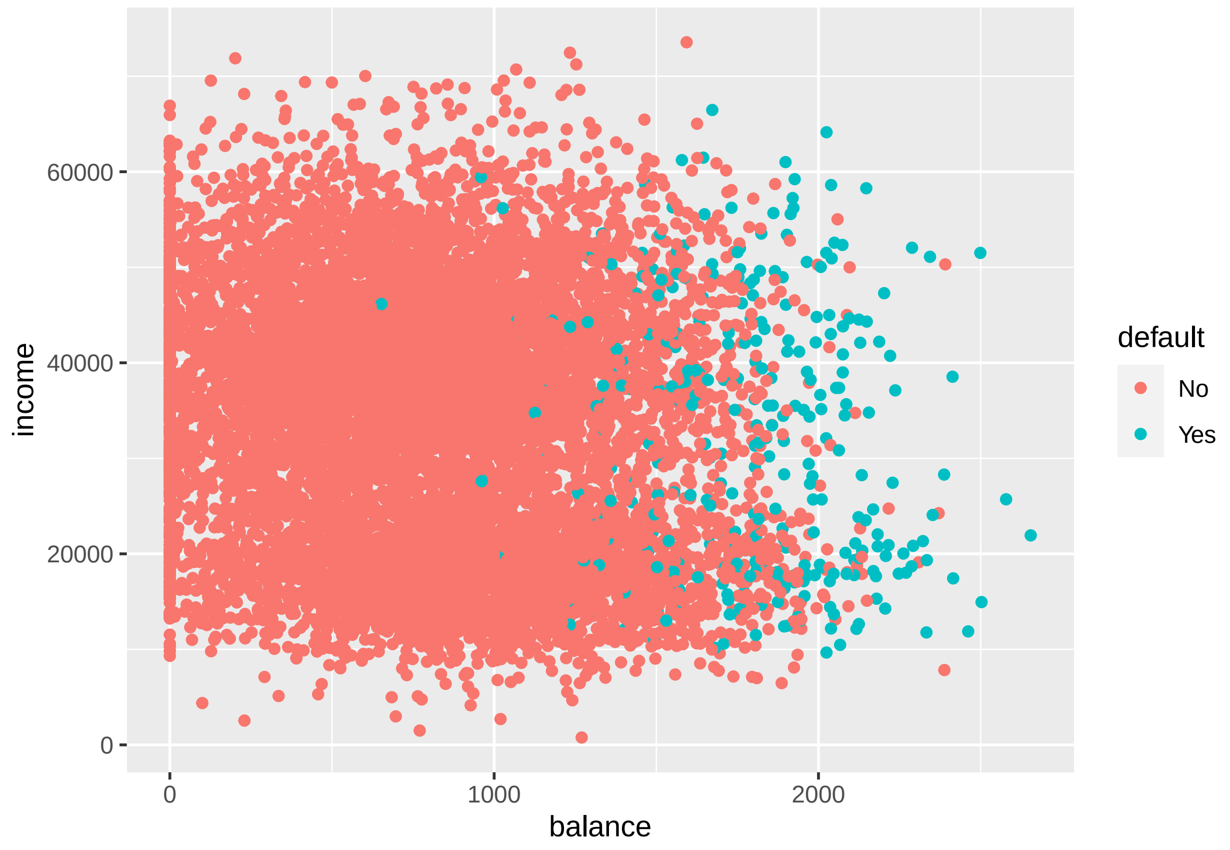
The Default dataset

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)
```

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

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



Overall, we see that most observations do not default, and those that do are more likely to have higher balances (>2000). We also see almost no defaults if the balance is less than 1000.

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

```
set.seed(12345)

default.train.tbl <- default.tbl%>%
  slice_sample(n = 8000)

default.test.tbl <- setdiff(default.tbl, default.train.tbl)
```

6. Create a linear model (similar to point 4) that predicts `default` based on `balance` and `income`. What is the missclassification rate?

```
#Assigning 0 and 1 to default variable
default.train.tbl <- default.train.tbl%>%
  mutate(isDefault = ifelse(default == "Yes", 1, 0))

#Modeling and Predicting
default.lm <- lm(isDefault ~ balance + income, data = default.train.tbl)
summary(default.lm)
```

```
##
## Call:
```

```
## lm(formula = isDefault ~ balance + income, data = default.train.tbl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24217 -0.06778 -0.02640  0.01927  0.98621
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.491e-02  6.411e-03 -14.805  < 2e-16 ***
## balance      1.284e-04  3.893e-06  32.977  < 2e-16 ***
## income       5.988e-07  1.414e-07   4.235 2.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1667 on 7997 degrees of freedom
## Multiple R-squared:  0.1197, Adjusted R-squared:  0.1195
## F-statistic: 543.9 on 2 and 7997 DF,  p-value: < 2.2e-16
```

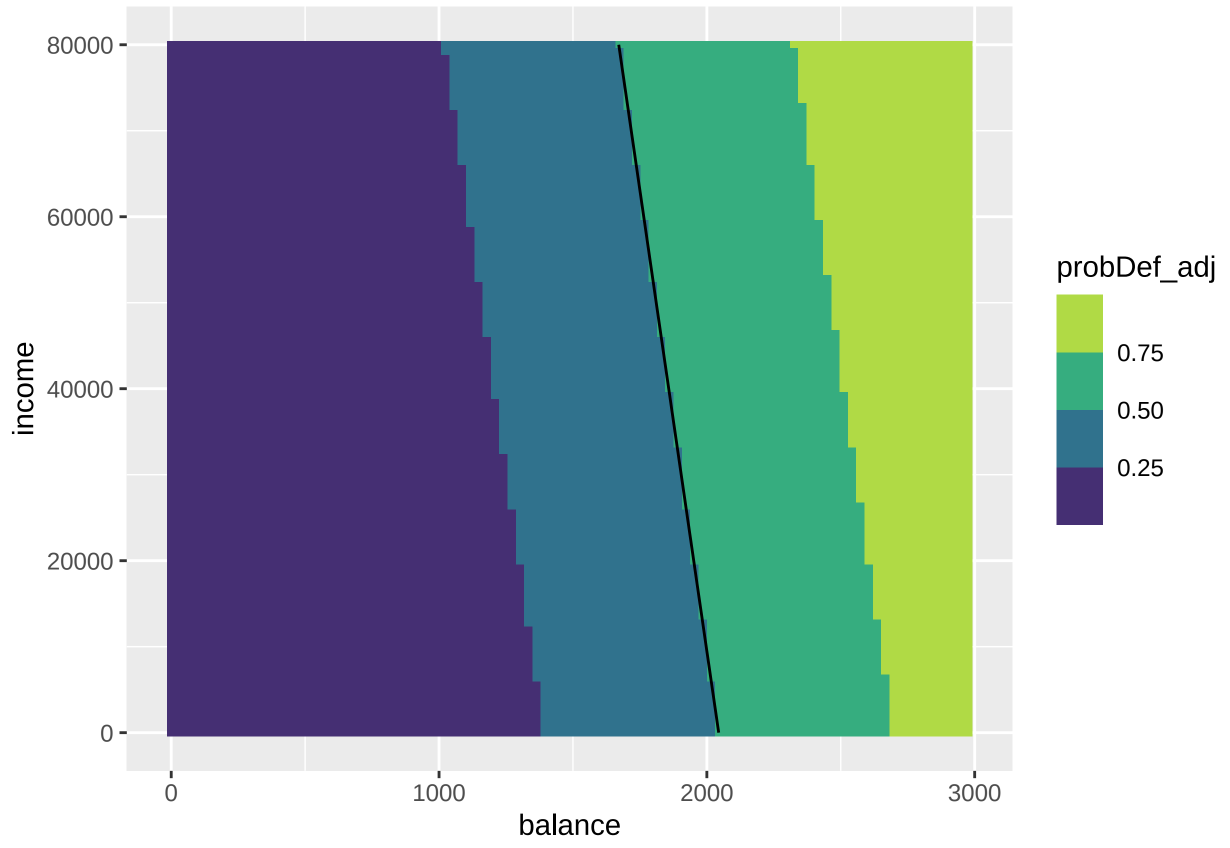
```r
prob1 <- predict(default.lm, default.test.tbl)

#Multiply by 1/max(probDef) --> ensures max value is now 1

default.test.tbl <- default.test.tbl%>%
  mutate(probDef = prob1,
         probDef = ifelse(probDef < 0, 0, probDef),
         probDef = ifelse(probDef > 1, 1, probDef),
         probDef_adj = probDef * (1/max(probDef)),
         predDef = ifelse(probDef_adj >=0.5, "Yes", "No"))


#Misclassification Rate

mean(default.test.tbl$predDef != default.test.tbl$default)
```

```
## [1] 0.062
```

After adjusting our probabilities so that the maximum probability is calibrated as '1', we have a missclassification rate of 0.062.

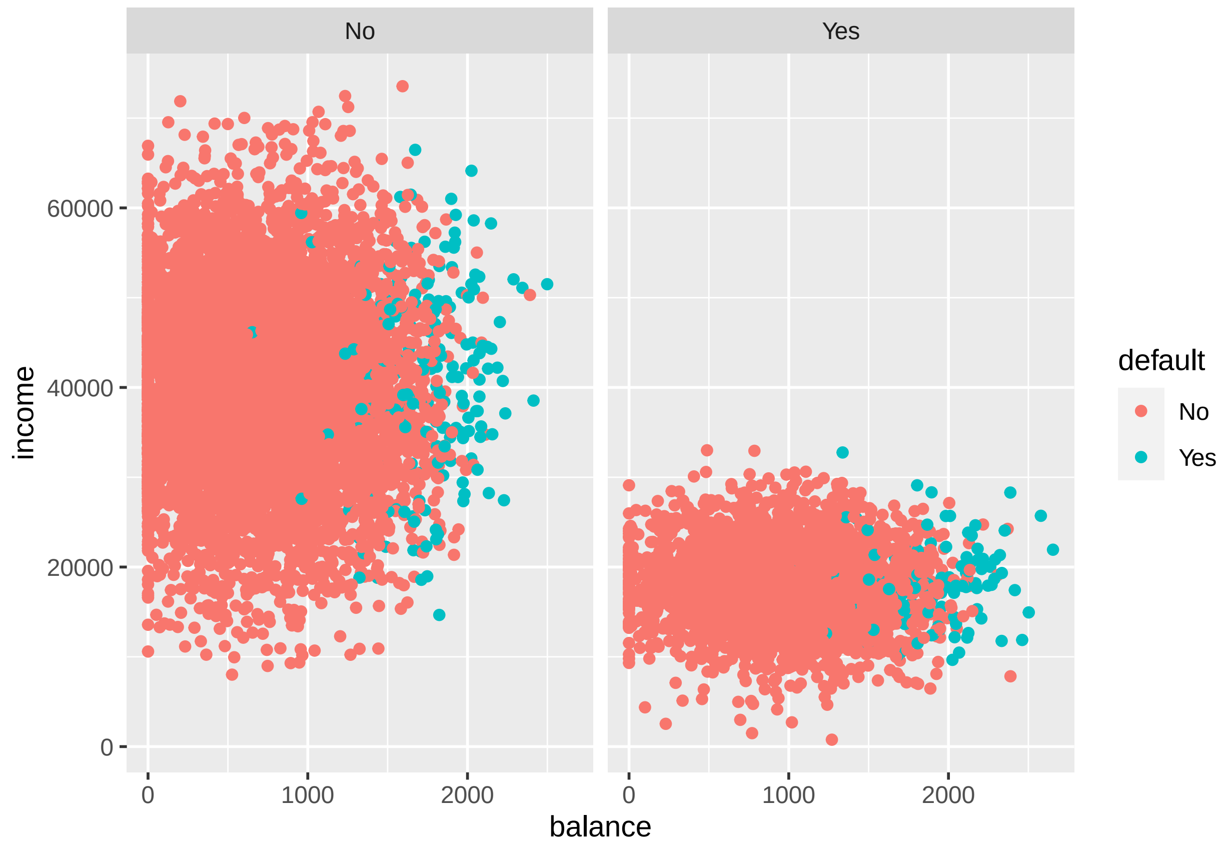
7. Plot the probability of the model created on 6) on a grid where $(x_1, x_2) \in [0, 3000] \times [0, 80000]$. Make sure your grid **does not have over 10,000 points**. Plot the decision boundary of the model as well.

```r
#Make grid - (stepped by 31 so that the expanded grid was < 10000)
vec1 <-seq(0, 3000, 31)
vec2 <-seq(0, 80000, 800)

def.grid.tbl <- expand_grid(balance = vec1, income = vec2)

prob2 <- predict(default.lm, def.grid.tbl)

#Same as before - calibrating data
def.grid.tbl <- def.grid.tbl%>%
  mutate(probDef = prob2,
         probDef = ifelse(probDef < 0, 0, probDef),
         probDef = ifelse(probDef > 1, 1, probDef),
         probDef_adj = probDef * (1/max(probDef)),
        predDef = ifelse(probDef_adj >=0.5, "Yes", "No"))

ggplot(def.grid.tbl, aes(balance, income, z=probDef_adj, fill = probDef_adj)) +
    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.

```
ggplot(default.tbl, aes(balance, income, color = default))+
  geom_point()+
  facet_wrap(~student, nrow = 1)
```



When we facet wrap based on whether the person is a student or not, we don't see much of a difference. As one would expect, the students have significantly lower income, but we see that those who default all have higher balances, regardless of student status.

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

```
#New Model
default2.lm <- lm(isDefault ~ balance + income + student, data = default.train.tbl)
summary(default2.lm)
```

```
##
## Call:
## lm(formula = isDefault ~ balance + income + student, data = default.train.tbl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24228 -0.06808 -0.02607  0.01937  0.98412
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.876e-02  9.282e-03  -8.486   <2e-16 ***
## balance      1.297e-04  3.931e-06  32.999   <2e-16 ***
## income       2.171e-07  2.126e-07   1.021   0.3071
## studentYes  -1.508e-02  6.270e-03  -2.405   0.0162 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1667 on 7996 degrees of freedom
## Multiple R-squared:  0.1204, Adjusted R-squared:   0.12
## F-statistic: 364.8 on 3 and 7996 DF,  p-value: < 2.2e-16
```

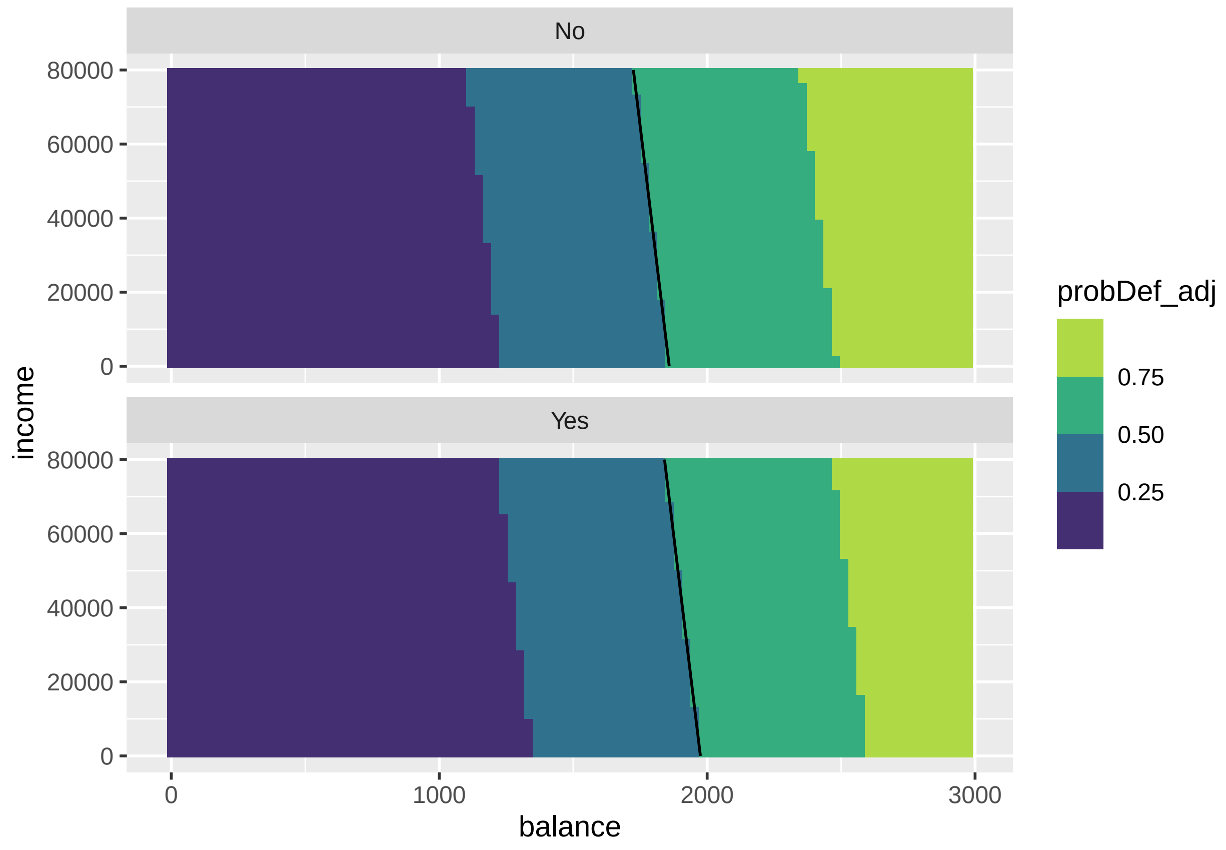
```
#Predicting values
prob2 <- predict(default2.lm, default.test.tbl)

#Same as before, calibrating values where max value is 1
default2.test.tbl <- default.test.tbl%>%
  mutate(probDef = prob2,
         probDef = ifelse(probDef < 0, 0, probDef),
         probDef = ifelse(probDef > 1, 1, probDef),
         probDef_adj = probDef * (1/max(probDef)),
         predDef = ifelse(probDef_adj >=0.5, "Yes", "No"))

#Misclassification Rate
mean(default2.test.tbl$predDef != default2.test.tbl$default)
```

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

When we consider student status in our model, we get a missclassification rate of 0.605, a missclassification rate that is 0.0015 lower than what we found without considering student status.

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

```
#Making grid as before, this time with third vector of student
vec1 <-seq(0, 3000, 31)
vec2 <-seq(0, 80000, 800)
vec3 <- factor(c("Yes", "No"))

def.grid2.tbl <- expand_grid(balance = vec1, income = vec2, student = vec3)

prob2 <- predict(default2.lm, def.grid2.tbl)

#Same as before, calibrating predictions
def.grid2.tbl <- def.grid2.tbl%>%
  mutate(probDef = prob2,
         probDef = ifelse(probDef < 0, 0, probDef),
         probDef = ifelse(probDef > 1, 1, probDef),
         probDef_adj = probDef * (1/max(probDef)),
         predDef = ifelse(probDef_adj >=0.5, "Yes", "No"))

#Raster plot faceted by student status
ggplot(def.grid2.tbl, aes(balance, income, z=probDef_adj, fill = probDef_adj)) +
    geom_raster() +
    stat_contour(breaks=c(0.5), color="black")+
    scale_fill_viridis_b()+
  facet_wrap(~student, nrow = 2)
```


No
Yes
80000
60000
40000
20000
0
income
0
1000
2000
3000
balance
probDef_adj
0.75
0.50
0.25