Classification

Jaime Davila

2/20/2022

The MNIST dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits used by the Machine learning community. The `dslabs` packages has a handy function called `read_mnist` that allows to load this dataset as follows:

```
mnist <- read_mnist("~/Mscs 341 S22/Class/Data")
```

The first thing to notice about this dataset is its structure which we can find using the function `str`

```
str(mnist)
```

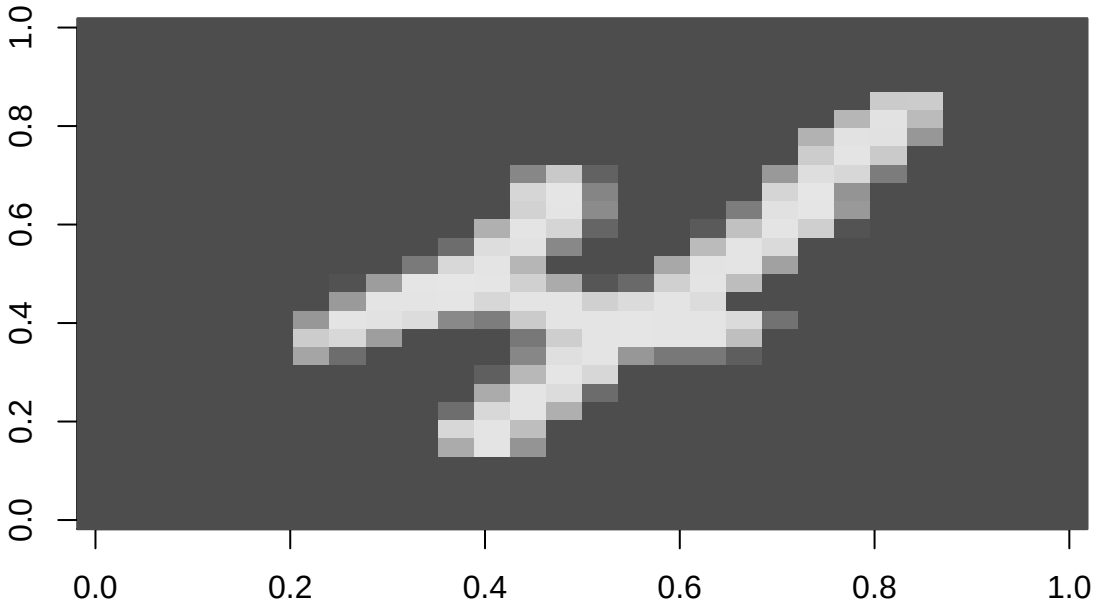
```
## List of 2
##  $ train:List of 2
##   ..$ images: int [1:60000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ labels: int [1:60000] 5 0 4 1 9 2 1 3 1 4 ...
##  $ test :List of 2
##   ..$ images: int [1:10000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ labels: int [1:10000] 7 2 1 0 4 1 4 9 5 9 ...
```

As we can wee `mnist` has a training and testing set. The training dataset has 60,000 elements represented as a matrix of 6000×784 (every image is a vector of 784, representing a 28×28 image). It also has the labels corresponding to each of the images represented as integers. Finally the testing dataset has 10,000 elements represented in a similar way. Before we interact with this dataset, let's define a handy function that will allows us to plot any digit:

```
plotImage <- function(dat,size=28){
  imag <- matrix(dat,nrow=size)[,28:1]
  image(imag,col=grey.colors(256), xlab = "", ylab="")
}
```

So now let's explore a couple of elements from our training and testing datasets

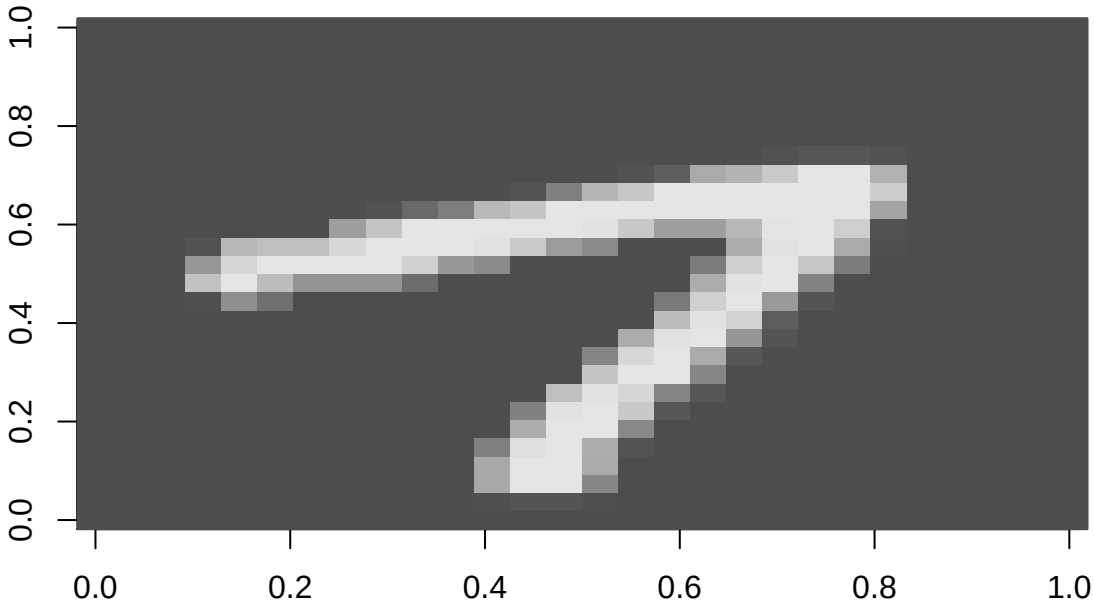
```
# We plot the 10th element from our training dataset which is a 4.
plotImage(mnist$train$images[10,])
```




```
mnist$train$labels[10]
```

```
## [1] 4
```

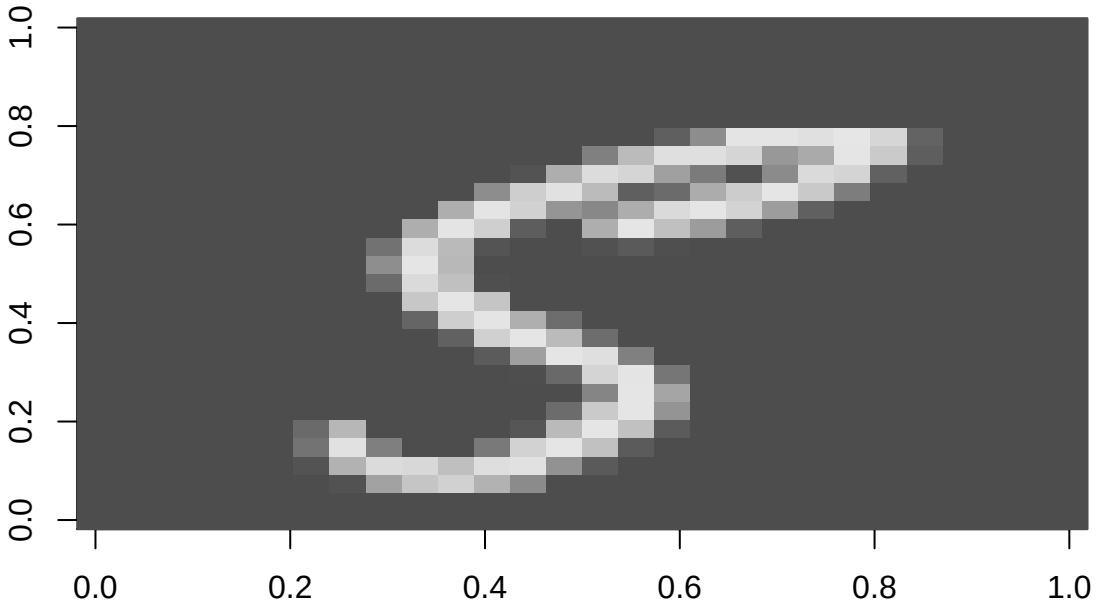
```
# We plot the 102th element from our training dataset which is a 7.
plotImage(mnist$train$images[102,])
```




```
mnist$train$labels[102]
```

```
## [1] 7
```

```
# We plot the 212th element from our testing dataset which is a 5.
plotImage(mnist$test$images[212,])
```




```
mnist$test$labels[212]
```

```
## [1] 5
```

Is it a 2 or a 7?

Our original problems has 784 predictors and a response variable with 10 different levels corresponding to each digit. We will start by simplifying our problem/dataset to recognize whether a digit is a 2 or a 7 and we will be using only two predictors, namely:

- `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.

Conveniently for us `dslabs` contains a random sample of 1000 digits (800 in training and 200 in testing). Let's load the dataset and plot it

```
data("mnist_27")
str(mnist_27)
```

```
## List of 5
##  $ train      :'data.frame': 800 obs. of  3 variables:
##   ..$ y  : Factor w/ 2 levels "2","7": 1 2 1 1 2 1 2 2 2 1 ...
##   ..$ x_1: num [1:800] 0.0395 0.1607 0.0213 0.1358 0.3902 ...
##   ..$ x_2: num [1:800] 0.1842 0.0893 0.2766 0.2222 0.3659 ...
##  $ test       :'data.frame': 200 obs. of  3 variables:
##   ..$ y  : Factor w/ 2 levels "2","7": 1 2 2 2 2 1 1 1 1 2 ...
##   ..$ x_1: num [1:200] 0.148 0.283 0.29 0.195 0.218 ...
```

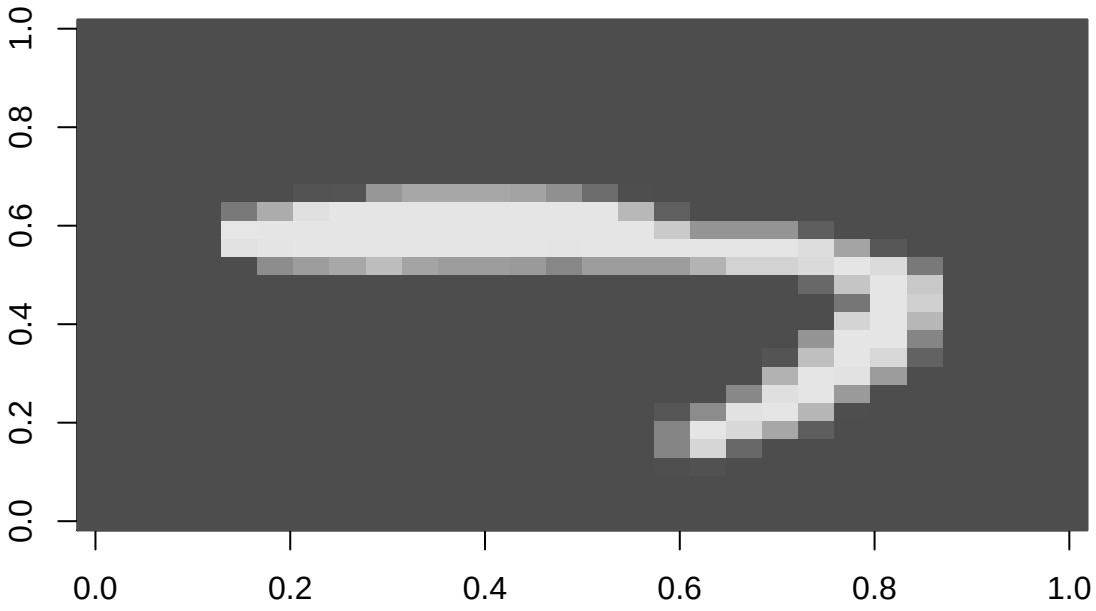
```
##   ..$ x_2: num [1:200] 0.261 0.348 0.435 0.115 0.397 ...
##  $ index_train: int [1:800] 40334 33996 3200 38360 36239 38816 8085 9098 15470 5096 ...
##  $ index_test : int [1:200] 46218 35939 23443 30466 2677 54248 5909 13402 11031 47308 ...
##  $ true_p     :'data.frame': 22500 obs. of  3 variables:
##   ..$ x_1: num [1:22500] 0 0.00352 0.00703 0.01055 0.01406 ...
##   ..$ x_2: num [1:22500] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ p  : num [1:22500] 0.703 0.711 0.719 0.727 0.734 ...
##   ..- attr(*, "out.attrs")=List of 2
##   .. ..$ dim     : Named int [1:2] 150 150
##   .. .. ..- attr(*, "names")= chr [1:2] "x_1" "x_2"
##   .. ..$ dimnames:List of 2
##   .. .. ..$ x_1: chr [1:150] "x_1=0.0000000" "x_1=0.0035155" "x_1=0.0070310" "x_1=0.0105465" ...
##   .. .. ..$ x_2: chr [1:150] "x_2=0.000000000" "x_2=0.004101417" "x_2=0.008202834" "x_2=0.012304251"
```

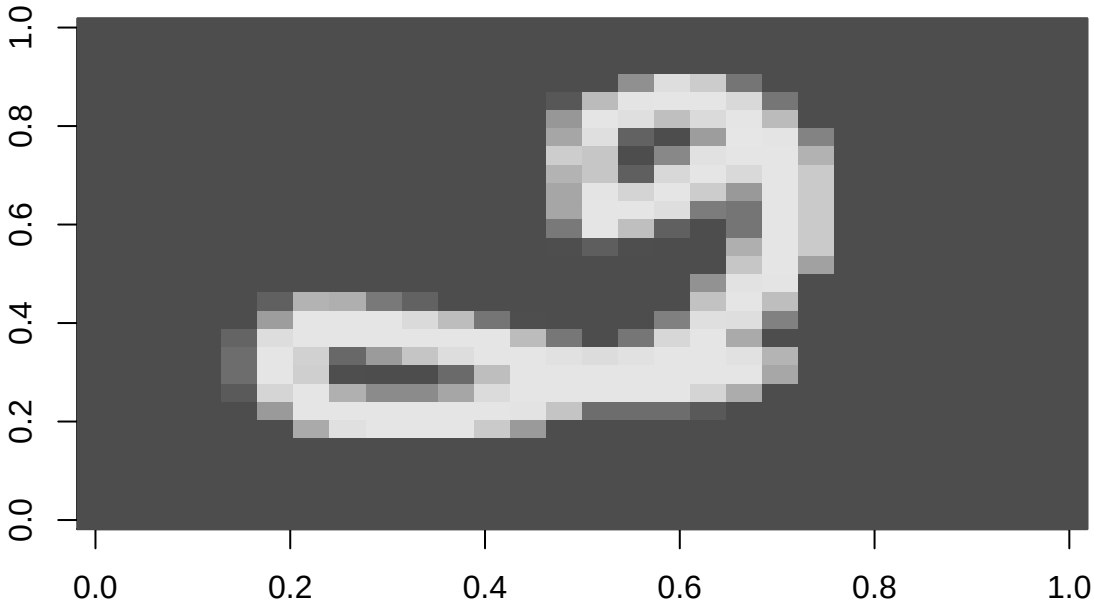
```
train.tbl <- tibble(mnist_27$train)
train.tbl <-  train.tbl %>%
  mutate(n=row_number())

test.tbl <- tibble(mnist_27$test)
test.tbl <-  test.tbl %>%
  mutate(n=row_number())

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

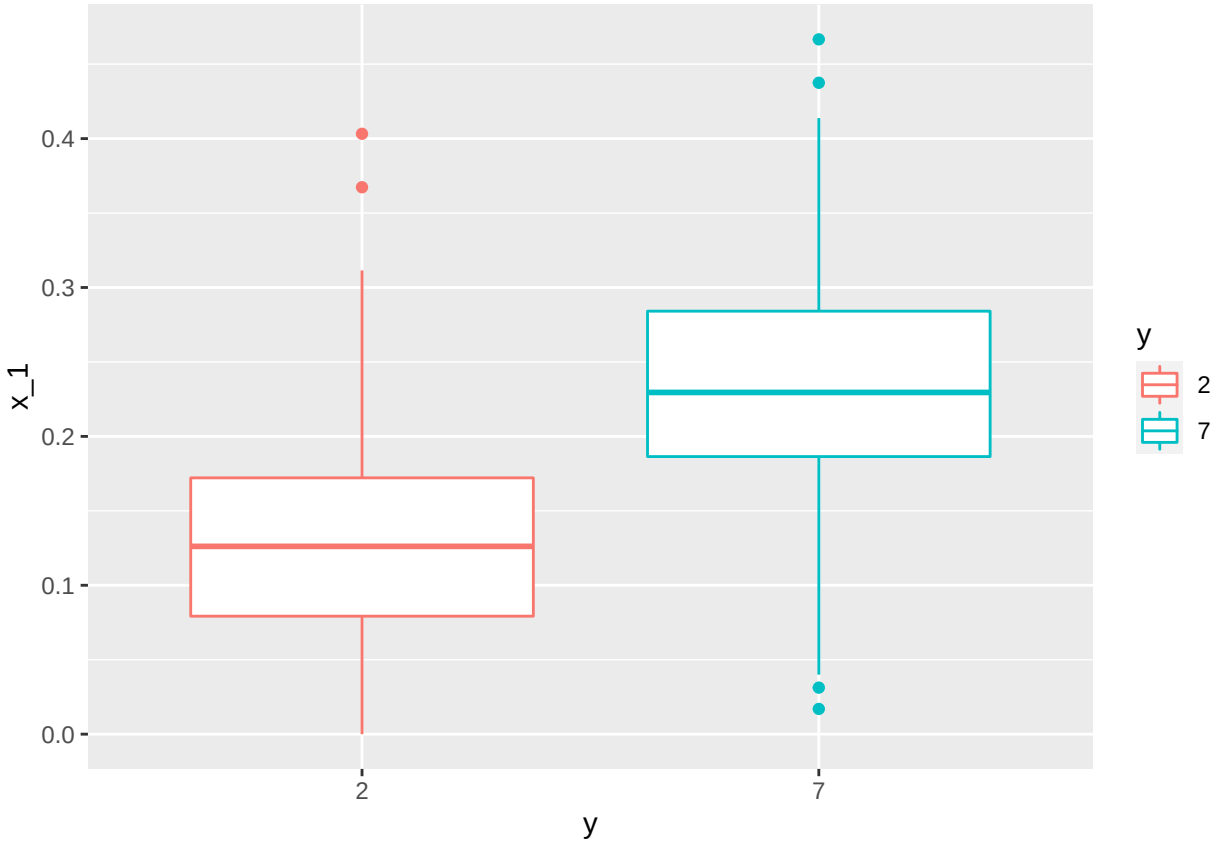
1. Pick the elements with the largest and smallest values in `x_1` and plot their corresponding images. Do a boxplot comparing the value of `x_1` across 2 and 7s. Does `x1` allow you to distinguish in general

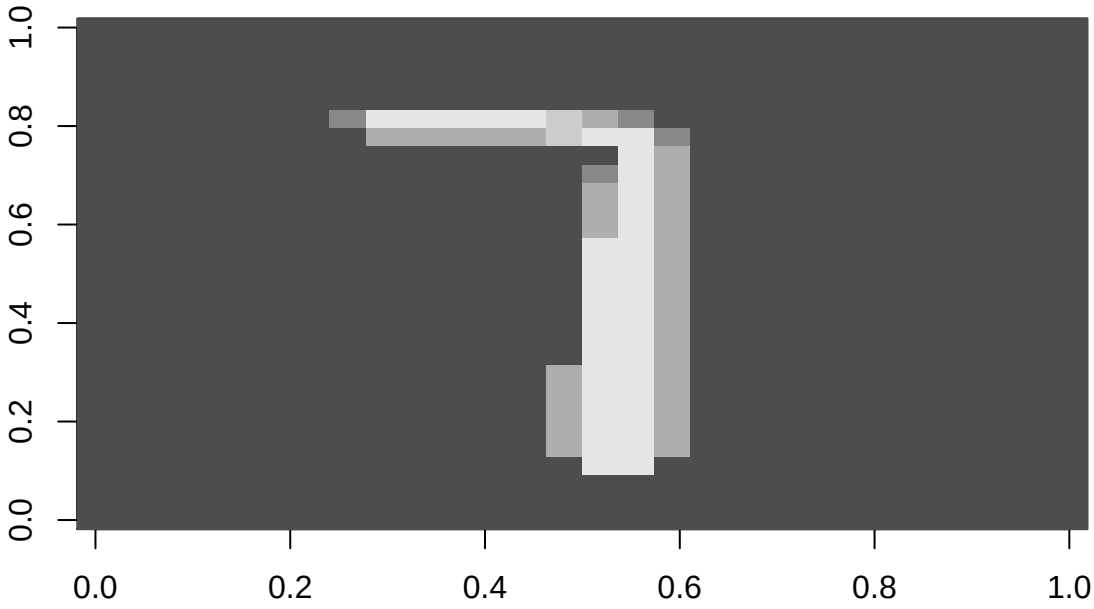
between a 2 and 7?. Do the same analysis for `x_2`.

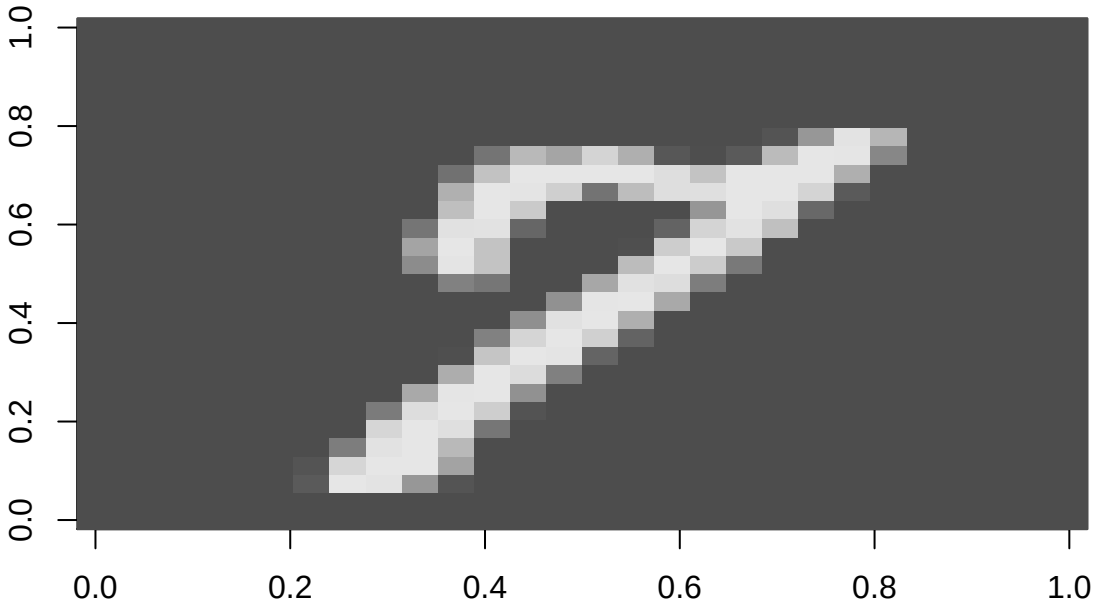
Hint:Notice that the corresponding index in the `mnist` dataset can be found by using `mnist_27$index_train`

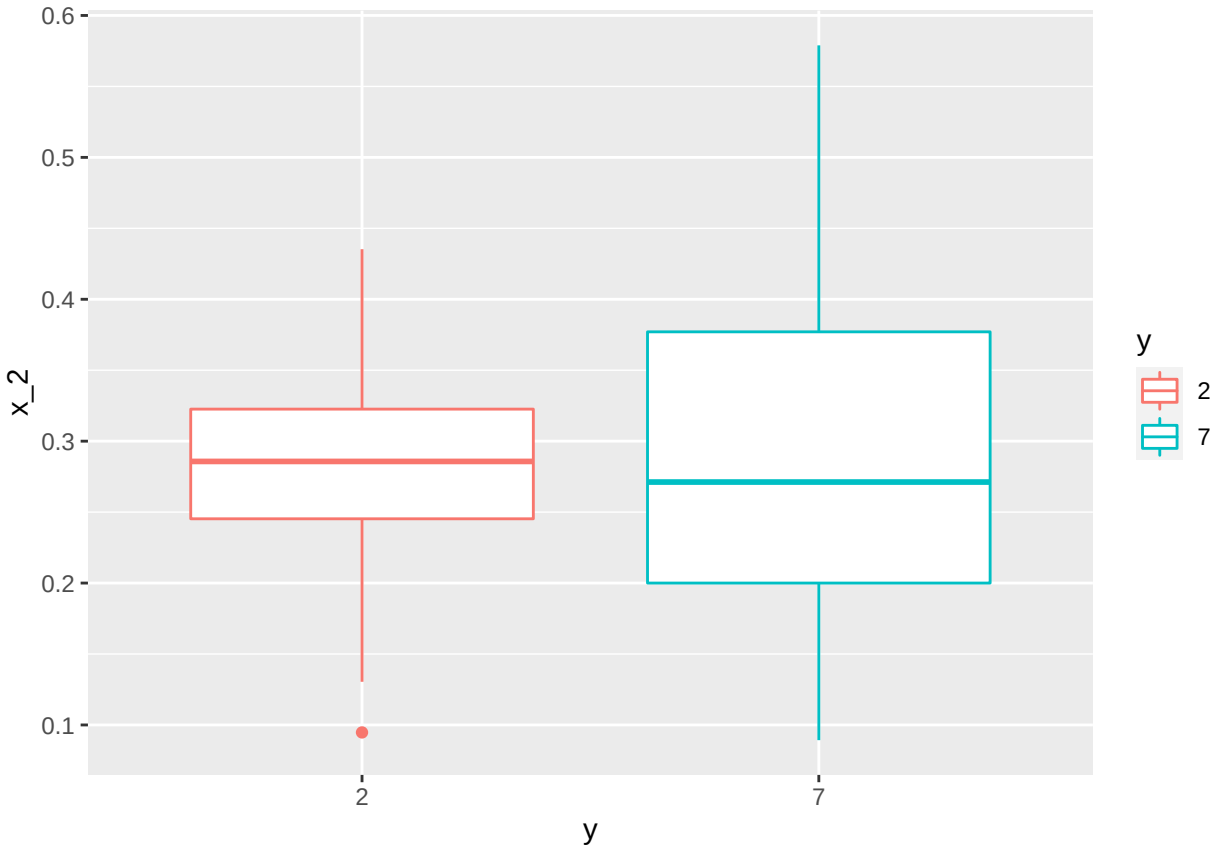



0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.4
0.3
0.2
0.1
0.0
x_1
2
7
y
y
2
7


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0



Classification and KNN

We can build our first model by using a KNN model. Notice that since we are in the classification setting we will be using the function `knn3`

```
kNear=5
knn.model <- knn3(y~x_1+x_2, data=train.tbl, k=kNear)
```

Notice that we can use the function `predict` on our testing dataset

```
pred.prob <- predict(knn.model, test.tbl)
head(pred.prob)
```

```
##        2   7
## [1,] 0.8 0.2
## [2,] 0.0 1.0
## [3,] 0.2 0.8
## [4,] 0.0 1.0
## [5,] 0.6 0.4
## [6,] 1.0 0.0
```

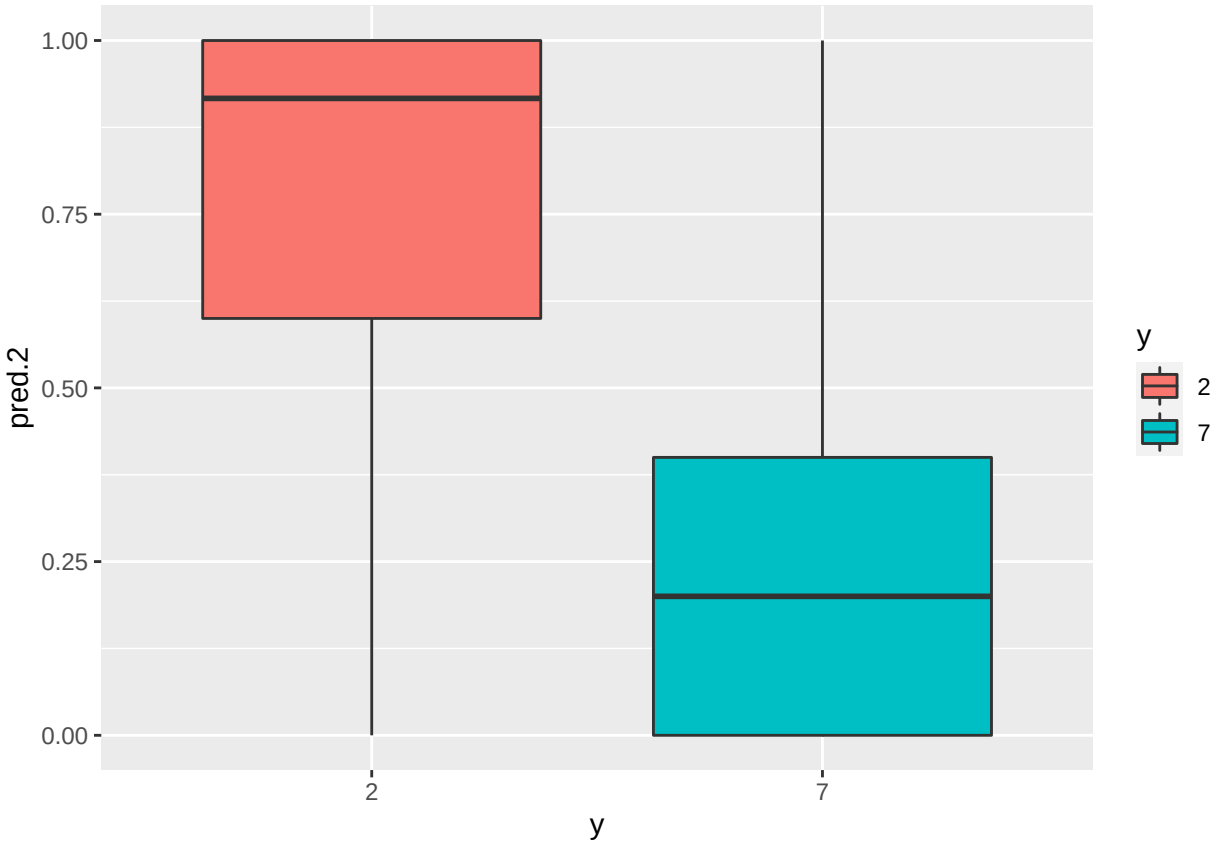
If we are interested in obtaining a class label we can do so by using the parameter `type="class"`

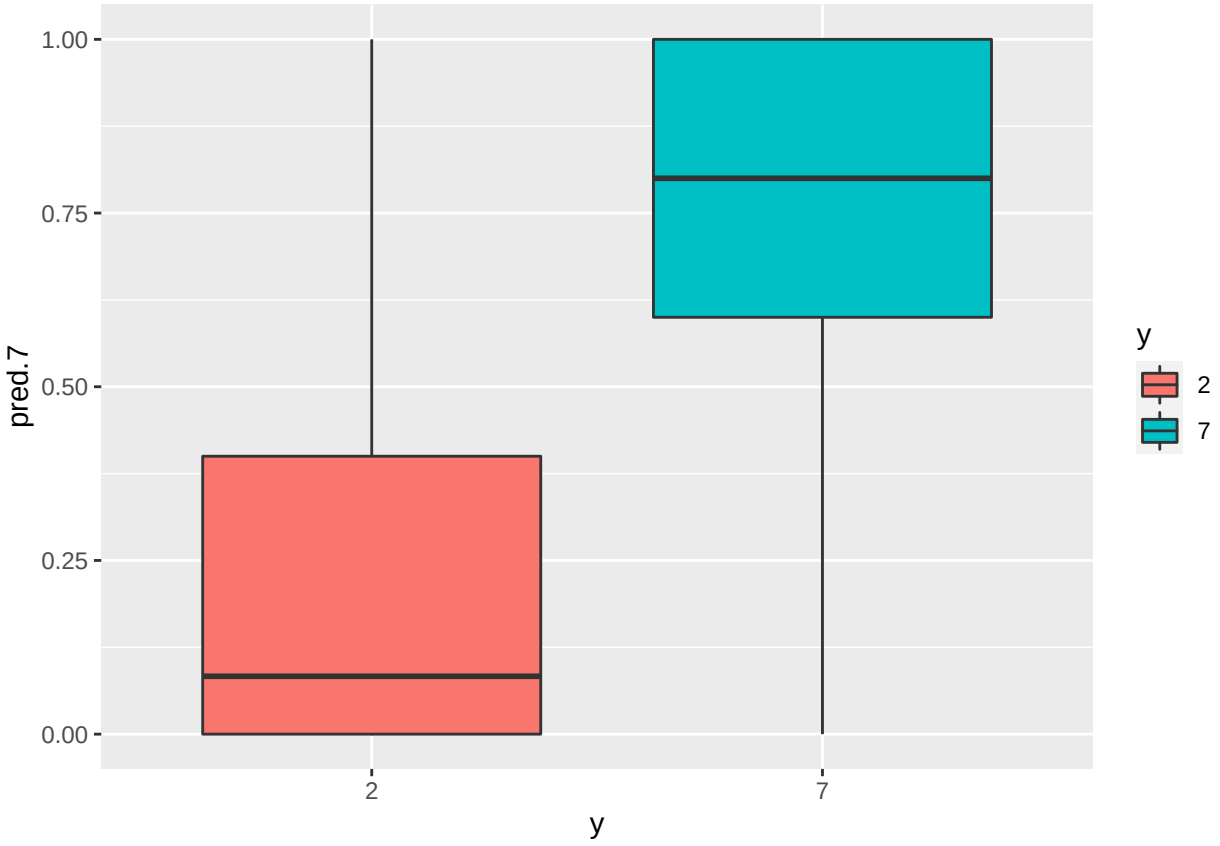
```
pred <- predict(knn.model, test.tbl, type="class")
head(pred)
```

```
## [1] 2 7 7 7 2 2
```

```
## Levels: 2 7
```

2. Plot the probability that an element is a 2 on the testing dataset





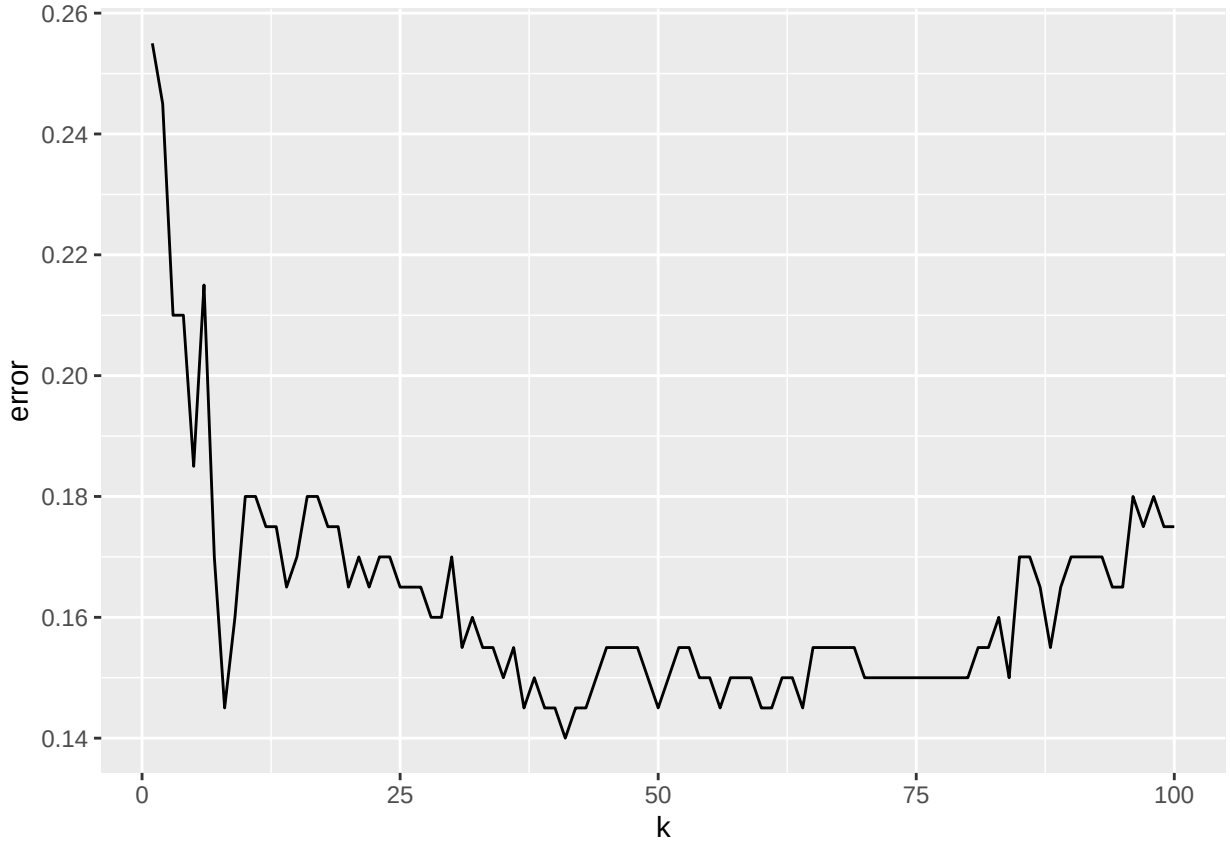
3. Create a function `calc_error (kNear, train, test)` that calculates the misclassification error for KNN using `knear` neighbors

```
## [1] 0.185
```

```
## [1] 0.145
```

```
## [1] 0.18
```

4. Plot the value of `k` against the misclassification error for $k = 1..100$ using the testing dataset. What is the optimal value of `k`?




```
## # A tibble: 1 x 2
##       k error
##   <int> <dbl>
## 1    41  0.14
```

5. (*Optional*) Using the optimal value of `k` identify cases that are missclassified. Is it more common that 2 gets confused for a 7 or the other way around? Plot a couple of digits that are missclassified. Any ideas for features that would allow to distinguish those?