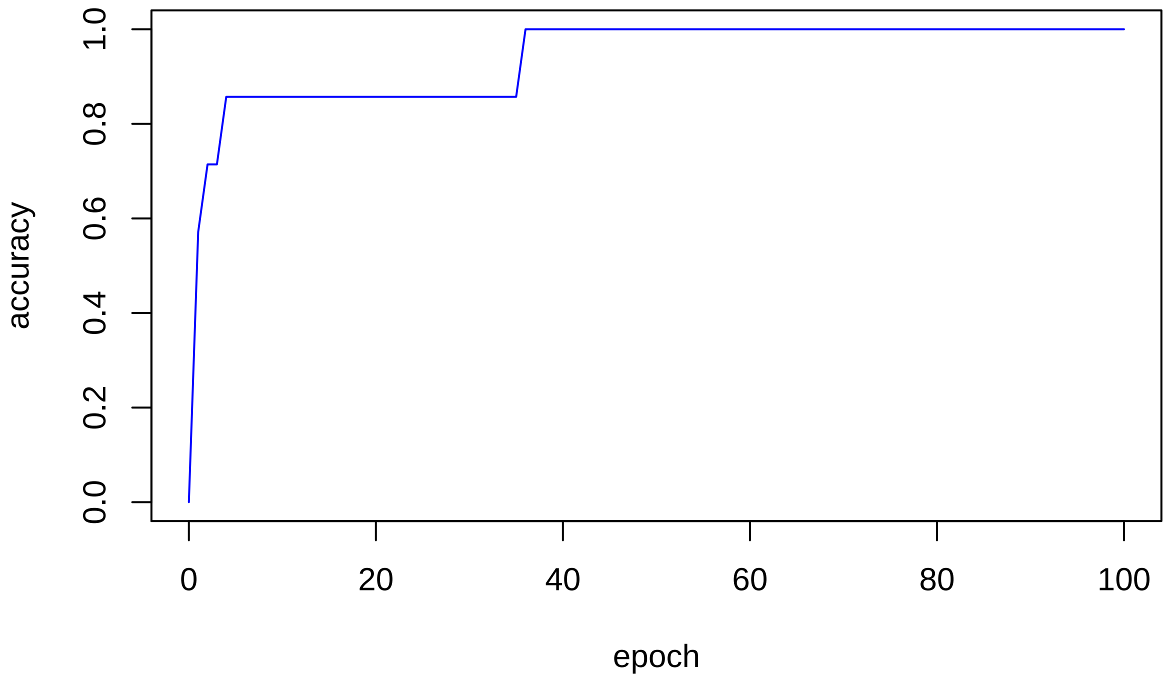
Homework2

Kai Wang

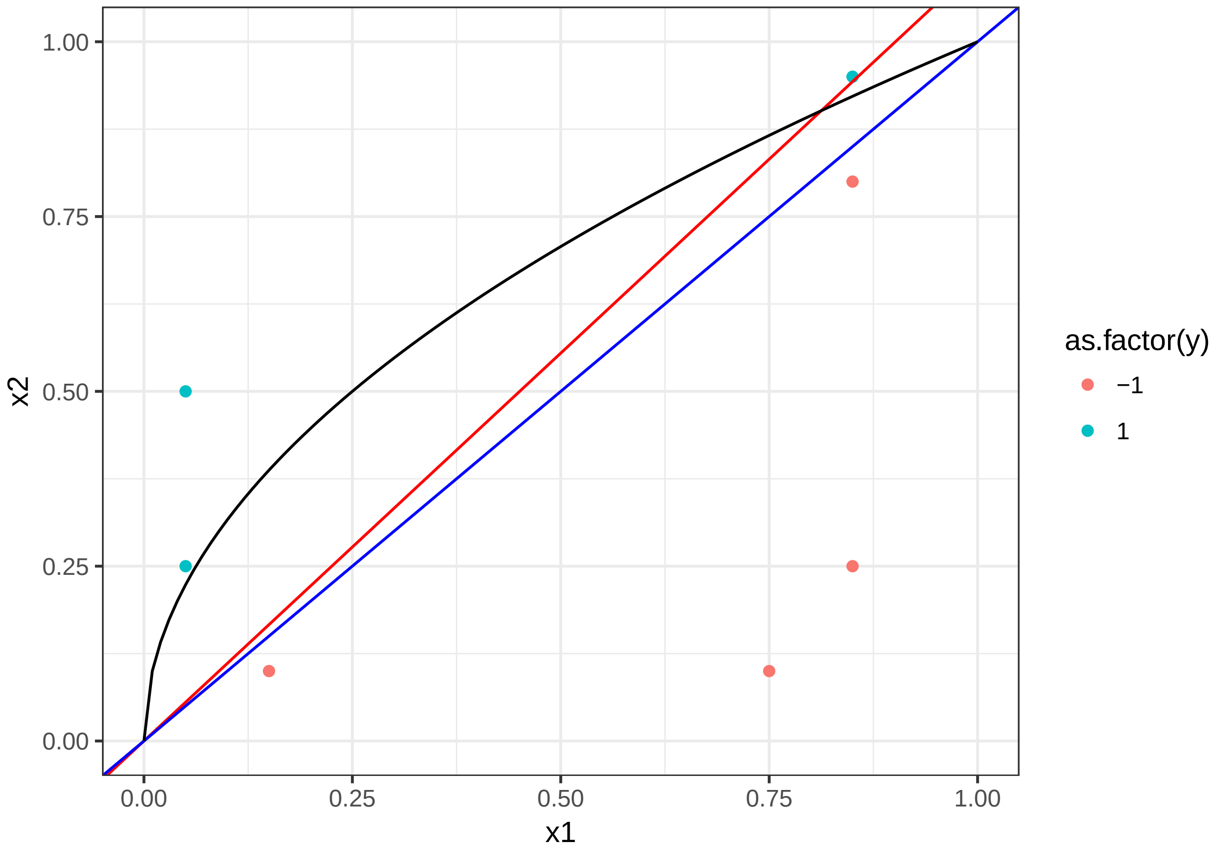
2018/1/30

1 Classifiers for Basketball Courts

a



From the epoch vs. accuracy plot, we can see the perceptron converges at 37th iteration. Since the accuracy is 100%, there's no empirical error for this classifier. We can come up with other linear classifiers which will give the same error(0). For example, an boolean function $y = f(x_1, x_2) = \mathbb{I}_{-x_1+x_2>0}$. More generally, as long as the slope of the linear classifier passing through origin is between $(\frac{16}{17}, \frac{19}{17})$.

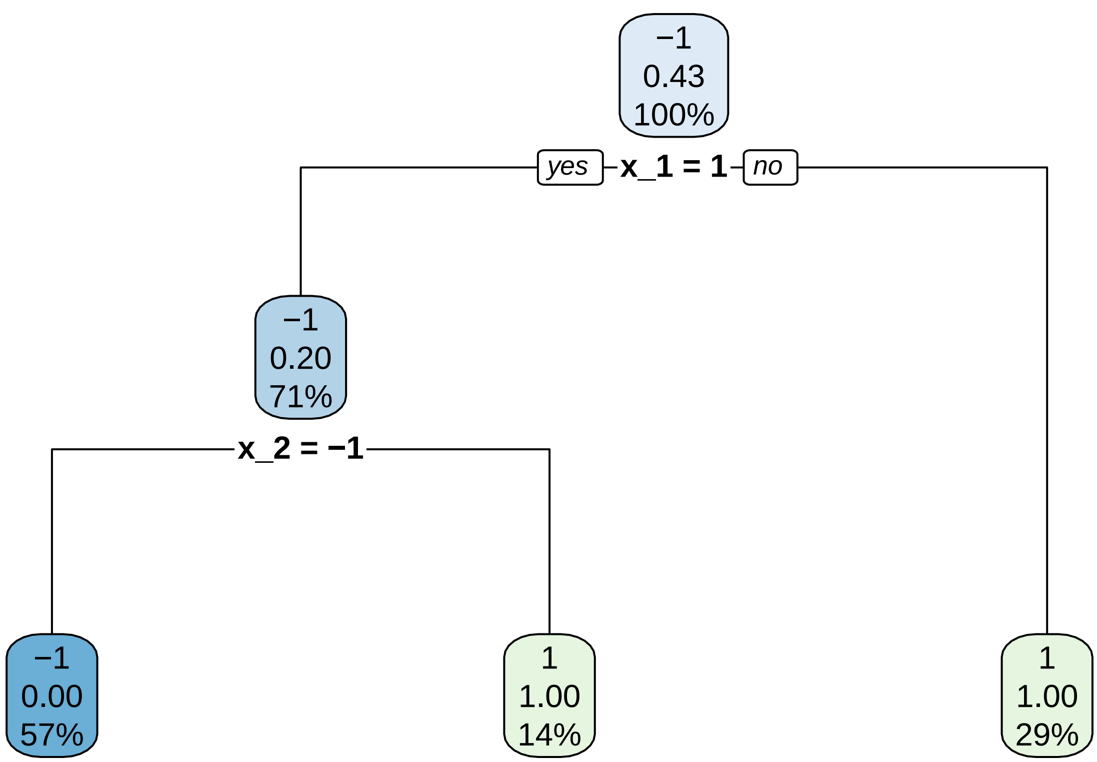


The above plot shows data and the decision boundary of the perceptron (as red line). In addition, the boolean function $y = f(x_1, x_2) = \mathbb{I}_{-x_1+x_2+0.08>0}$ is also plotted as a black line, which clearly seperates our data with no empirical error.

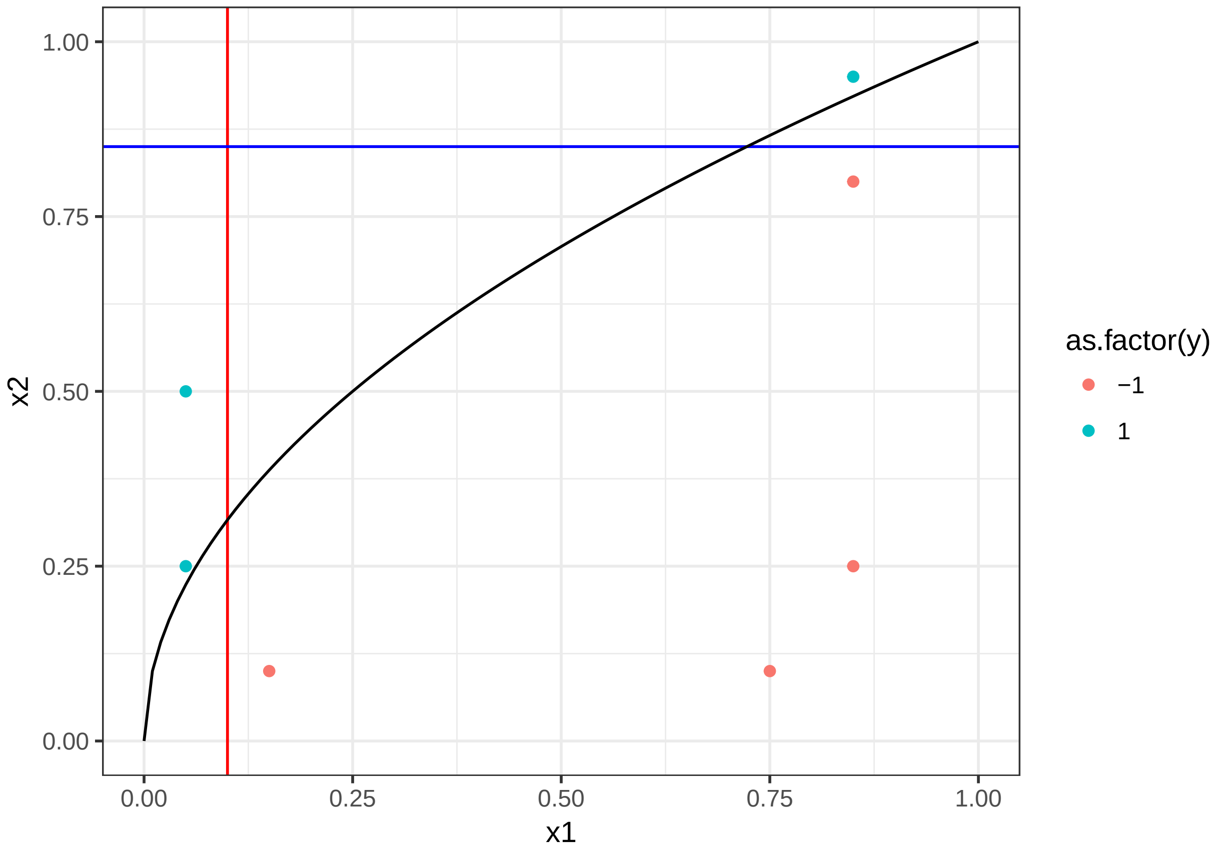
b

```
## Call:
## rpart(formula = as.factor(y) ~ x_1 + x_2, data = shots, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 1))
##   n= 7
##
##          CP nsplit rel error    xerror      xstd
## 1 0.6666667      0 1.0000000 1.0000000 0.4364358
## 2 0.3333333      1 0.3333333 1.0000000 0.4364358
## 3 0.0100000      2 0.0000000 0.3333333 0.3086067
##
## Variable importance
## x_1 x_2
##  53  47
##
## Node number 1: 7 observations,    complexity param=0.6666667
##   predicted class=-1  expected loss=0.4285714  P(node) =1
##     class counts:     4     3
##    probabilities: 0.571 0.429
##   left son=2 (5 obs) right son=3 (2 obs)
##   Primary splits:
```

```
##       x_1 splits as  RL, improve=1.8285710, (0 missing)
##       x_2 splits as  LR, improve=0.7619048, (0 missing)
##
## Node number 2: 5 observations,    complexity param=0.3333333
##   predicted class=-1  expected loss=0.2  P(node) =0.7142857
##     class counts:     4     1
##    probabilities: 0.800 0.200
##   left son=4 (4 obs) right son=5 (1 obs)
##   Primary splits:
##       x_2 splits as  LR, improve=1.6, (0 missing)
##
## Node number 3: 2 observations
##   predicted class=1   expected loss=0  P(node) =0.2857143
##     class counts:     0     2
##    probabilities: 0.000 1.000
##
## Node number 4: 4 observations
##   predicted class=-1  expected loss=0  P(node) =0.5714286
##     class counts:     4     0
##    probabilities: 1.000 0.000
##
## Node number 5: 1 observations
##   predicted class=1   expected loss=0  P(node) =0.1428571
##     class counts:     0     1
##    probabilities: 0.000 1.000
```



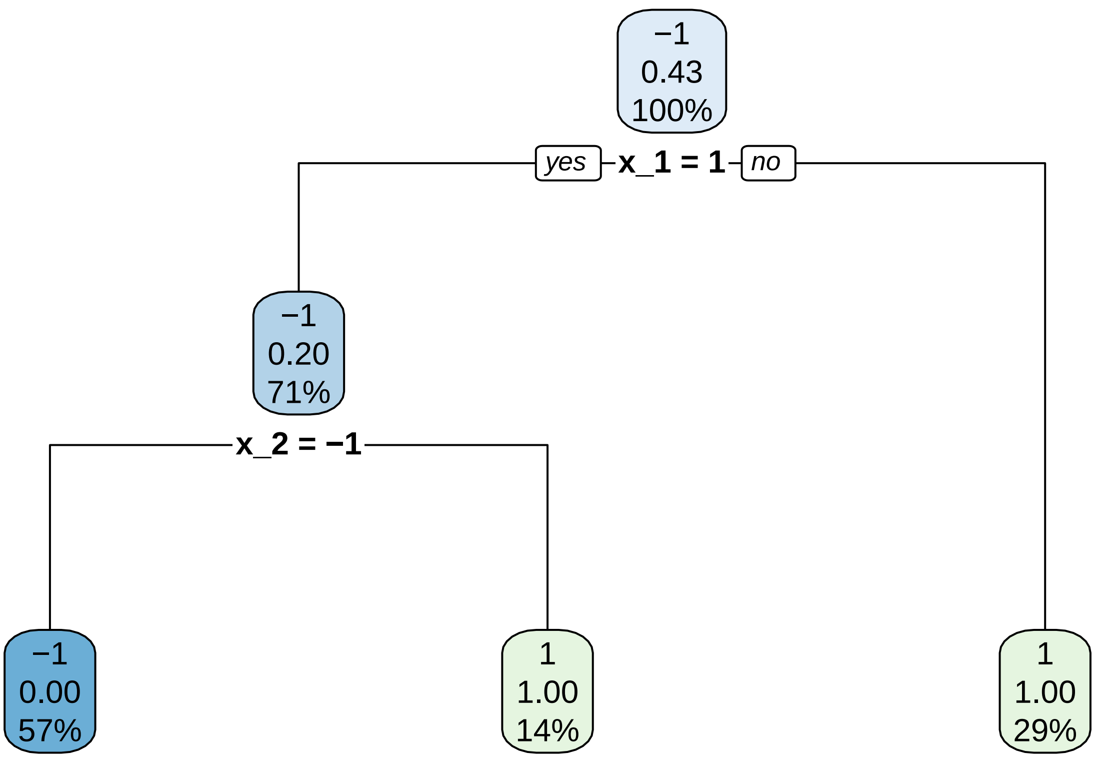
We can use "rpart" package to split our decision tree based on gini index. And this decision tree has all data points correctly classified.



We can choose $x_1 > 0.1$ and $x_2 > 0.85$ as our threshold for x_1, x_2, then split the decision tree based on transformed data. Using the reduction in the Gini index as the splitting criterion, the error is 0. We can adjust the threshold for x_1 between $[0.05, 0.15)$ and for x_2 bewtween $[0.80, 0.95)$, which will give us the same split on the data. The decision tree of such a threshold is plotted as follow.

```
## Call:
## rpart(formula = as.factor(y) ~ x_1 + x_2, data = shots, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 1))
##   n= 7
##
##          CP nsplit rel error    xerror      xstd
## 1 0.6666667      0 1.0000000 1.0000000 0.4364358
## 2 0.3333333      1 0.3333333 1.0000000 0.4364358
## 3 0.0100000      2 0.0000000 0.3333333 0.3086067
##
## Variable importance
## x_1 x_2
##  53  47
##
## Node number 1: 7 observations,    complexity param=0.6666667
##   predicted class=-1  expected loss=0.4285714  P(node) =1
##     class counts:     4     3
##    probabilities: 0.571 0.429
##   left son=2 (5 obs) right son=3 (2 obs)
##   Primary splits:
##       x_1 splits as  RL, improve=1.8285710, (0 missing)
##       x_2 splits as  LR, improve=0.7619048, (0 missing)
```

```
##
## Node number 2: 5 observations,    complexity param=0.3333333
##   predicted class=-1  expected loss=0.2  P(node) =0.7142857
##     class counts:     4     1
##    probabilities: 0.800 0.200
##   left son=4 (4 obs) right son=5 (1 obs)
##   Primary splits:
##       x_2 splits as  LR, improve=1.6, (0 missing)
##
## Node number 3: 2 observations
##   predicted class=1   expected loss=0  P(node) =0.2857143
##     class counts:     0     2
##    probabilities: 0.000 1.000
##
## Node number 4: 4 observations
##   predicted class=-1  expected loss=0  P(node) =0.5714286
##     class counts:     4     0
##    probabilities: 1.000 0.000
##
## Node number 5: 1 observations
##   predicted class=1   expected loss=0  P(node) =0.1428571
##     class counts:     0     1
##    probabilities: 0.000 1.000
```



c

Since the three-point line is our approximation is $x_2 = \sqrt{x_1}$, we know $f(x)^{\text{true}} = \text{sign}(x_2 - \sqrt{x_1})$. We need to use a linear classifier that goes through origin, say $f(\mathbf{x}) = \text{sign}(x_2 - ax_1)$.

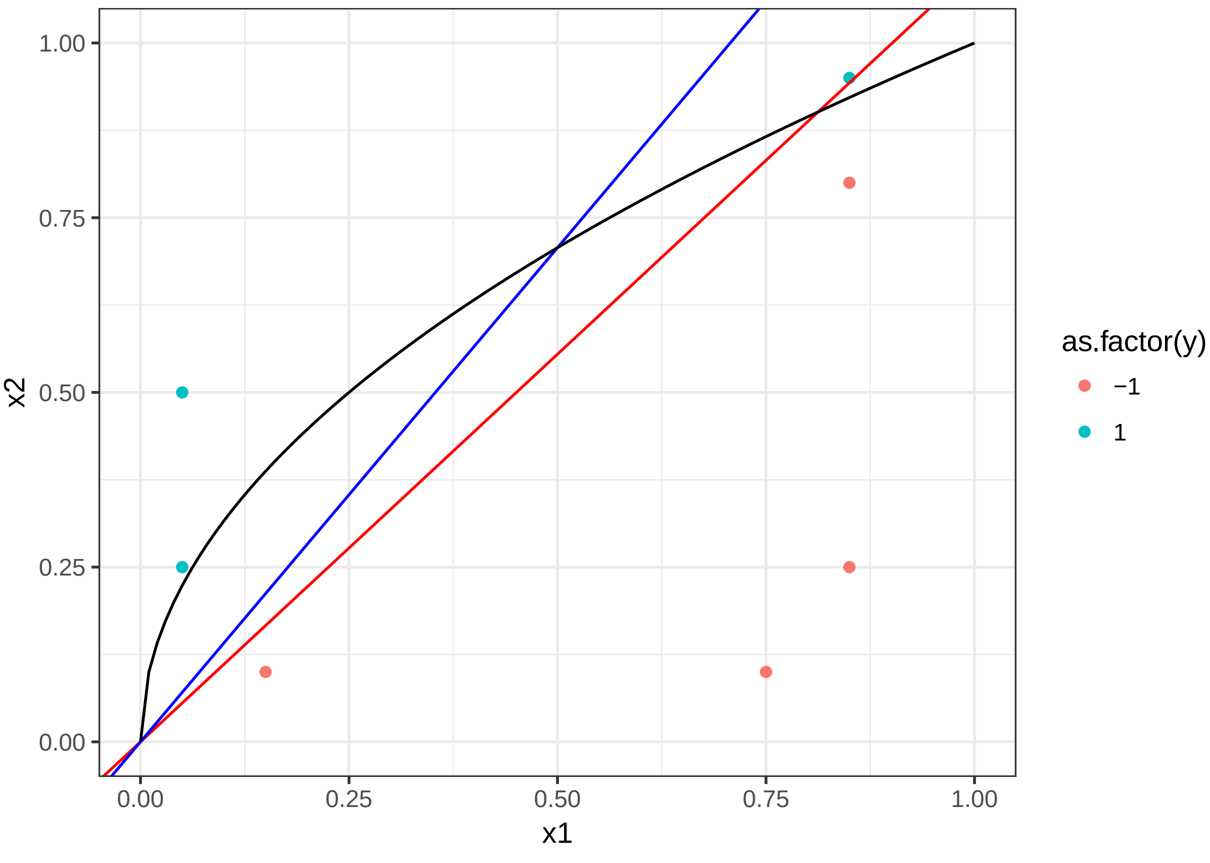
The true risk is $R^{\text{true}}(f) = \mathbb{E}_{(x,y)\sim D} l(f(\mathbf{x}), y)$, with the misclassification function $l(f(\mathbf{x}), y) = 1_{\text{sign}(f(\mathbf{x}) \neq y)}$.

Since $x_1, x_2 \sim \text{Uniform}[0,1]$, we transform this problem into finding the definite integral of area between $x_2 = \sqrt{x_1}$ and $x_2 = ax_1$, and it's intuitive that $a \geq 1$.

$$
\begin{aligned}
R^{\text{true}}(f) &= \int_0^{\frac{1}{a^2}} (\sqrt{x_1} - ax_1)dx_1 + \int_{\frac{1}{a^2}}^{\frac{1}{a}} (ax_1 - \sqrt{x_1})dx_1 + \int_{\frac{1}{a}}^{1} (1 - \sqrt{x_1})dx_1 \\
&= \frac{2}{3}a^{-3} - \frac{a}{2}a^{-4} + \frac{a}{2}a^{-2} - \frac{a}{2}a^{-4} - \frac{2}{3}a^{-\frac{3}{2}} + \frac{2}{3}a^{-3} + 1 - \frac{1}{a} - \frac{2}{3} + \frac{2}{3}a^{-\frac{3}{2}} \\
&= \frac{1}{3}a^{-3} - \frac{1}{2}a^{-1} + \frac{1}{3}
\end{aligned}
$$

To minimize the true risk, we need to choose a value of a so that $R^{\text{true}}(f)$ is minimized.

We have $\frac{dR^{\text{true}}(f)}{da} = \frac{1}{2a^2} - \frac{1}{a^4}$, which equals to 0 when $a = \sqrt{2}$. The second derivative shows that $R^{\text{true}}(f)$ is increasing after $a = \sqrt{2}$. Hence, $R^{\text{true}}(f)_{\min} = \frac{1}{3} - \frac{\sqrt{2}}{6}$.

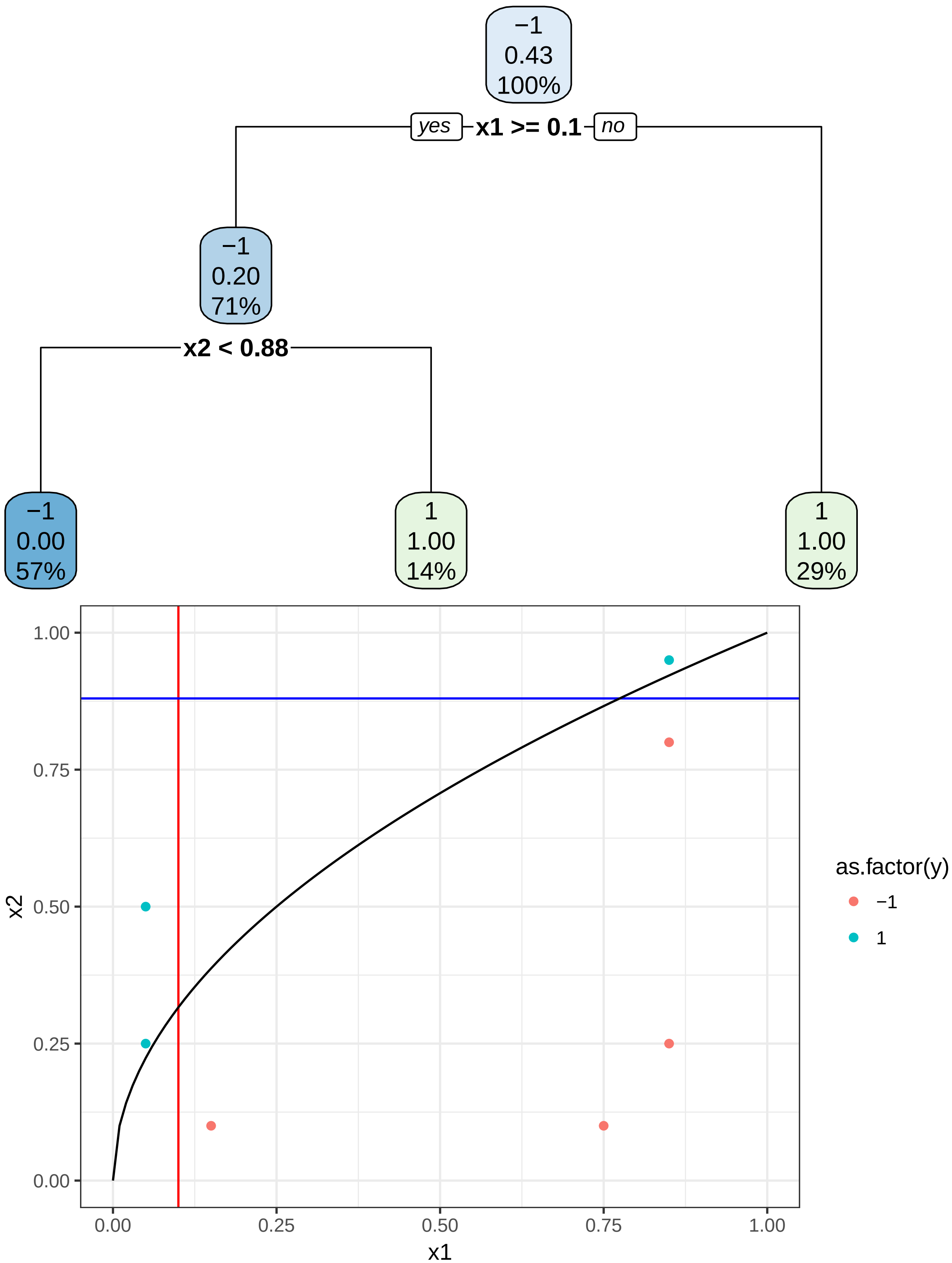


The blue line shows the optimal linear classifier, but empirically it classified 1 point wrong. The empirical error $R(f) = \frac{1}{8}$. Any linear classifier with slope between $(\frac{2}{3}, 5)$ would result in the same empirical error.

d

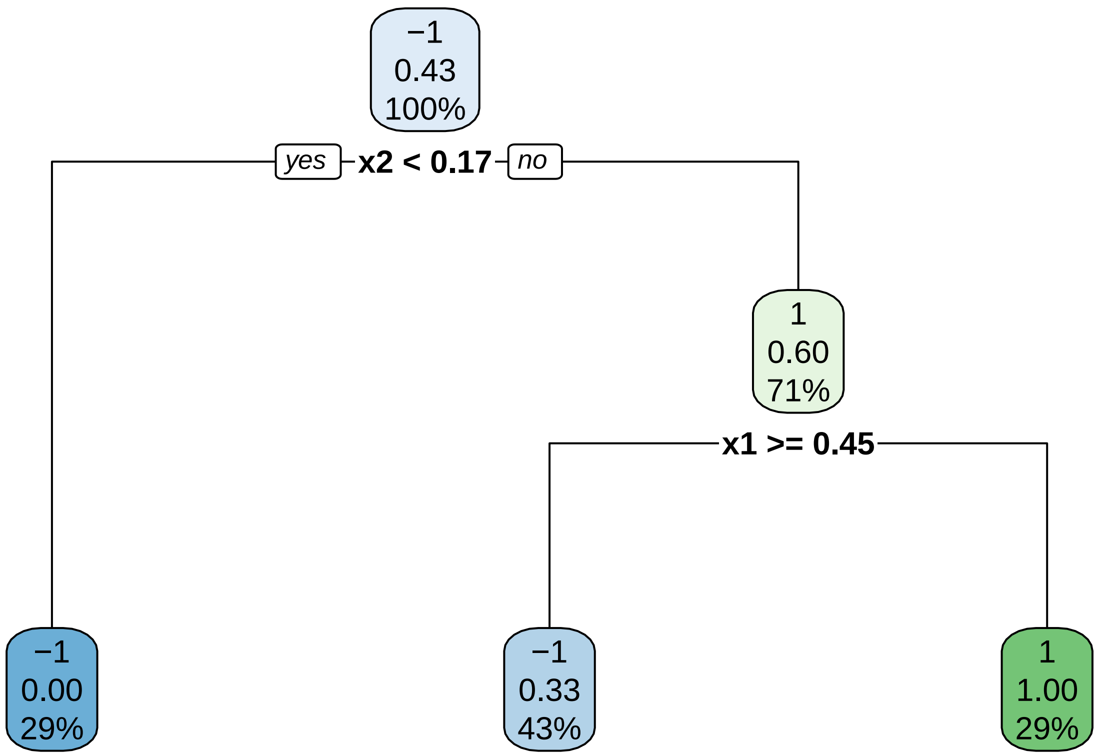
```
## Call:
## rpart(formula = as.factor(y) ~ x1 + x2, data = shots, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 1))
##   n= 7
```

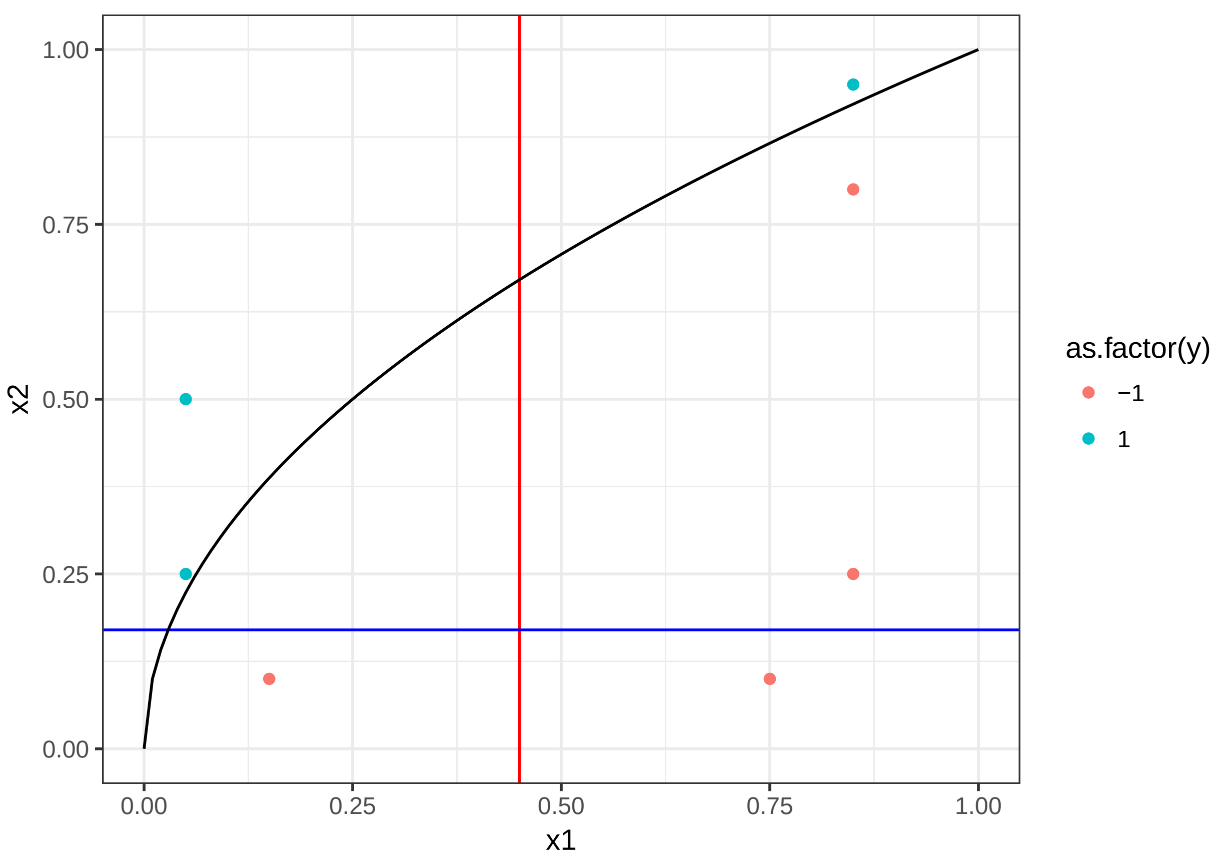
```
##
##          CP nsplit rel error   xerror      xstd
## 1 0.6666667      0 1.0000000 1.000000 0.4364358
## 2 0.3333333      1 0.3333333 1.333333 0.4364358
## 3 0.0100000      2 0.0000000 1.333333 0.4364358
##
## Variable importance
## x1 x2
## 53 47
##
## Node number 1: 7 observations,    complexity param=0.6666667
##   predicted class=-1  expected loss=0.4285714  P(node) =1
##     class counts:     4     3
##    probabilities: 0.571 0.429
##   left son=2 (5 obs) right son=3 (2 obs)
##   Primary splits:
##       x1 < 0.1   to the right, improve=1.828571, (0 missing)
##       x2 < 0.175 to the left,  improve=1.028571, (0 missing)
##
## Node number 2: 5 observations,    complexity param=0.3333333
##   predicted class=-1  expected loss=0.2  P(node) =0.7142857
##     class counts:     4     1
##    probabilities: 0.800 0.200
##   left son=4 (4 obs) right son=5 (1 obs)
##   Primary splits:
##       x2 < 0.875 to the left,  improve=1.6000000, (0 missing)
##       x1 < 0.8   to the left,  improve=0.2666667, (0 missing)
##
## Node number 3: 2 observations
##   predicted class=1   expected loss=0  P(node) =0.2857143
##     class counts:     0     2
##    probabilities: 0.000 1.000
##
## Node number 4: 4 observations
##   predicted class=-1  expected loss=0  P(node) =0.5714286
##     class counts:     4     0
##    probabilities: 1.000 0.000
##
## Node number 5: 1 observations
##   predicted class=1   expected loss=0  P(node) =0.1428571
##     class counts:     0     1
##    probabilities: 0.000 1.000
```



First decision tree splits on x_1 first and the x_2, we can see $s_1 = 0.1$ and $s_3 = 0.88$. s_2 does not exist since there is no split with $x_1 <= 0.1$. This is among the solutions that achieved the minimum empirical error.

```
## Call:
## rpart(formula = as.factor(y) ~ x1 + x2, data = shots, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 1,
##         maxdepth = 2), cost = c(2, 1))
##   n= 7
##
##          CP nsplit rel error   xerror      xstd
## 1 0.3333333      0 1.0000000 1.000000 0.4364358
## 2 0.0100000      2 0.3333333 1.666667 0.3984095
##
## Variable importance
## x2 x1
## 71 29
##
## Node number 1: 7 observations,    complexity param=0.3333333
##   predicted class=-1  expected loss=0.4285714  P(node) =1
##     class counts:     4     3
##    probabilities: 0.571 0.429
##   left son=2 (2 obs) right son=3 (5 obs)
##   Primary splits:
##       x2 < 0.175 to the left,  improve=1.0285710, (0 missing)
##       x1 < 0.1   to the right, improve=0.9142857, (0 missing)
##
## Node number 2: 2 observations
##   predicted class=-1  expected loss=0  P(node) =0.2857143
##     class counts:     2     0
##    probabilities: 1.000 0.000
##
## Node number 3: 5 observations,    complexity param=0.3333333
##   predicted class=1   expected loss=0.4  P(node) =0.7142857
##     class counts:     2     3
##    probabilities: 0.400 0.600
##   left son=6 (3 obs) right son=7 (2 obs)
##   Primary splits:
##       x1 < 0.45  to the right, improve=0.5333333, (0 missing)
##       x2 < 0.875 to the left,  improve=0.4000000, (0 missing)
##   Surrogate splits:
##       x2 < 0.65  to the right, agree=0.8, adj=0.5, (0 split)
##
## Node number 6: 3 observations
##   predicted class=-1  expected loss=0.3333333  P(node) =0.4285714
##     class counts:     2     1
##    probabilities: 0.667 0.333
##
## Node number 7: 2 observations
##   predicted class=1   expected loss=0  P(node) =0.2857143
##     class counts:     0     2
##    probabilities: 0.000 1.000
```

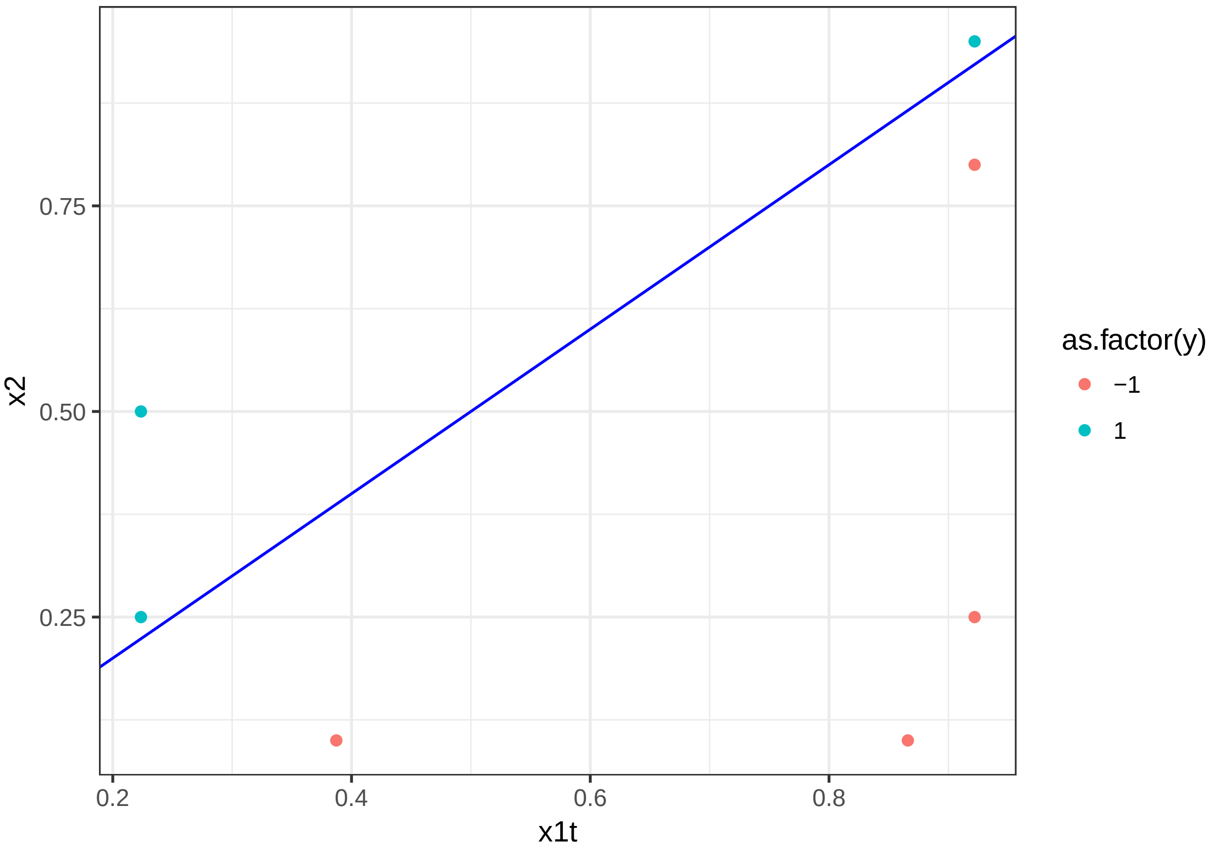




Second decision tree splits on x_2 first and the x_1, we can see $s_1 = 0.17$ and $s_2 = 0.45$. s_3 does not exist since there is no split with $x_2 <= 0.17$. This is not among the solutions that achieved the minimum empirical error.

e

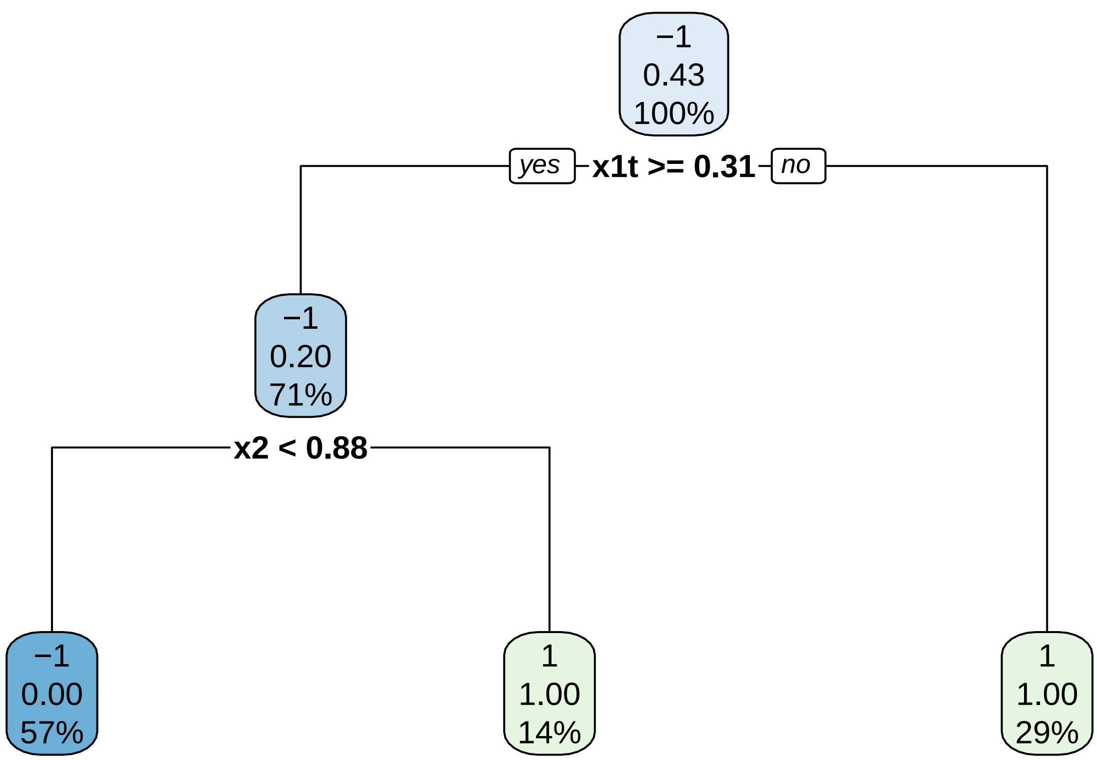
Since we know the true function $f(\mathbf{x}) = \mathbb{I}_{x_2 - \sqrt{x_1} > 0}$, we can make a tranformation $x_2 = \sqrt{x_1}$. Then our optimal linear classifier will simply be $y = \mathbb{I}_{x_2 - x_1 > 0}$ and its error is 0.

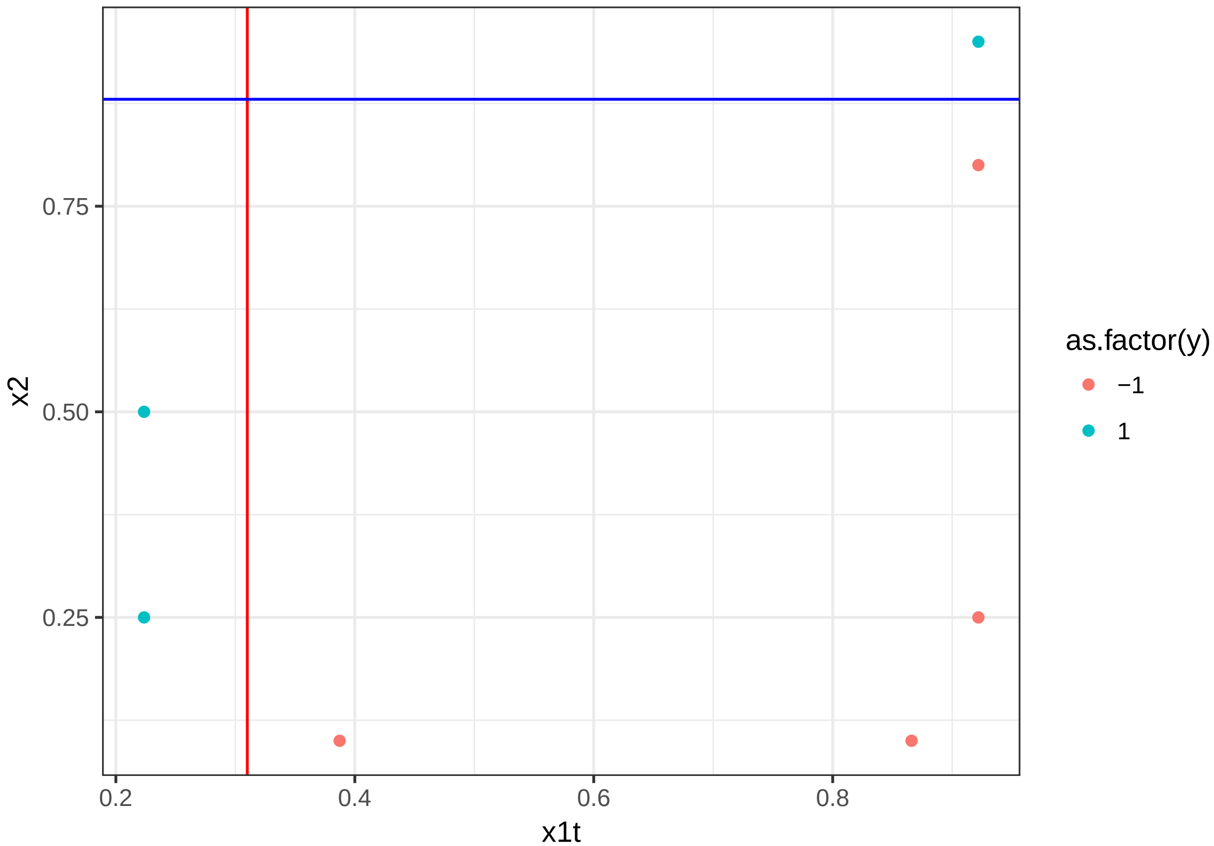


f

```
## Call:
## rpart(formula = as.factor(y) ~ x1t + x2, data = shots, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 1))
##   n= 7
##
##          CP nsplit rel error   xerror      xstd
## 1 0.6666667      0 1.0000000 1.000000 0.4364358
## 2 0.3333333      1 0.3333333 1.333333 0.4364358
## 3 0.0100000      2 0.0000000 1.333333 0.4364358
##
## Variable importance
## x1t  x2
##  53  47
##
## Node number 1: 7 observations,    complexity param=0.6666667
##   predicted class=-1  expected loss=0.4285714  P(node) =1
##     class counts:     4     3
##    probabilities: 0.571 0.429
##   left son=2 (5 obs) right son=3 (2 obs)
```

```
##   Primary splits:
##       x1t < 0.3054526 to the right, improve=1.828571, (0 missing)
##       x2  < 0.175     to the left,  improve=1.028571, (0 missing)
##
## Node number 2: 5 observations,    complexity param=0.3333333
##   predicted class=-1  expected loss=0.2  P(node) =0.7142857
##     class counts:     4     1
##    probabilities: 0.800 0.200
##   left son=4 (4 obs) right son=5 (1 obs)
##   Primary splits:
##       x2  < 0.875     to the left,  improve=1.6000000, (0 missing)
##       x1t < 0.8939899 to the left,  improve=0.2666667, (0 missing)
##
## Node number 3: 2 observations
##   predicted class=1   expected loss=0  P(node) =0.2857143
##     class counts:     0     2
##    probabilities: 0.000 1.000
##
## Node number 4: 4 observations
##   predicted class=-1  expected loss=0  P(node) =0.5714286
##     class counts:     4     0
##    probabilities: 1.000 0.000
##
## Node number 5: 1 observations
##   predicted class=1   expected loss=0  P(node) =0.1428571
##     class counts:     0     1
##    probabilities: 0.000 1.000
```





After the transformation on x_1, the decision tree achieved the same error.

h

We can represent the paint as area where $x_1 \geq 0.5, x_2 \leq 0.25$.

The true risk is $R^{\text{true}}(f) = \mathbb{E}_{(x,y)\sim D} l(f(\mathbf{x}), y)$, with the misclassification function $l(f(\mathbf{x}), y) = 1_{\text{sign}(f(\mathbf{x}) \neq y)}$.

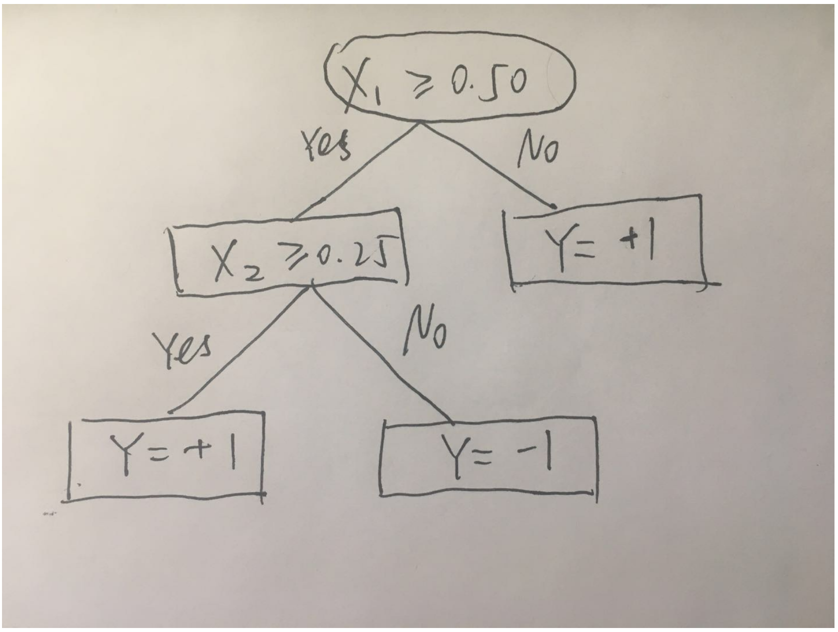
Since $x_1, x_2 \sim \text{Uniform}[0,1]$, we transform this problem into finding the definite integral of area between $x_2 = \begin{cases} 0.25, & x_1 \geq 0.5 \\ 0, & x_2 < 0.5 \end{cases}$ and $x_2 = ax_1$, and it's intuitive that $a \leq \frac{1}{2}$.

$$
\begin{aligned}
R^{\text{true}}(f) &= \int_0^{\frac{1}{2}} ax_1 dx_1 + \int_{\frac{1}{2}}^{\frac{1}{4a}} (\frac{1}{4} - ax_1) dx_1 + \int_{\frac{1}{4a}}^{1} (ax_1 - 1) dx_1 \\
&= \frac{3}{4}a + \frac{1}{4a} - \frac{9}{8}
\end{aligned}
$$

To minimize $R^{\text{true}}(f)$, we have $\frac{dR^{\text{true}}(f)}{da} = \frac{3}{4} - \frac{1}{4a^2} = 0$, which means $a = \frac{\sqrt{3}}{3}$. Since $R^{\text{true}}(f)$ is decreasing on $[\frac{1}{2}, \frac{\sqrt{3}}{3}]$ and increasing afterwards, we have $R^{\text{true}}(f)_{\min} = \frac{\sqrt{3}}{2} - \frac{9}{8}$.

i

The optimal decision tree will have decision boundaries that replicates the shape of paint and the error is 0.

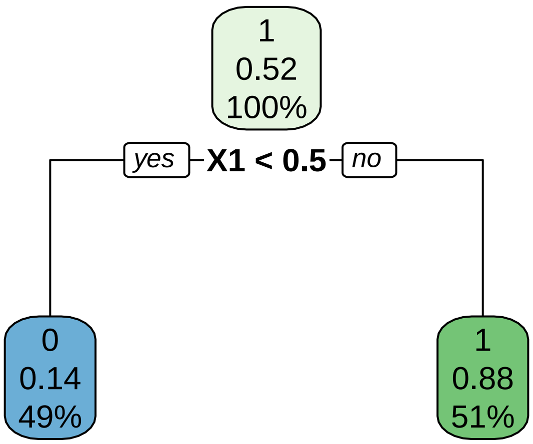


2

a(i)

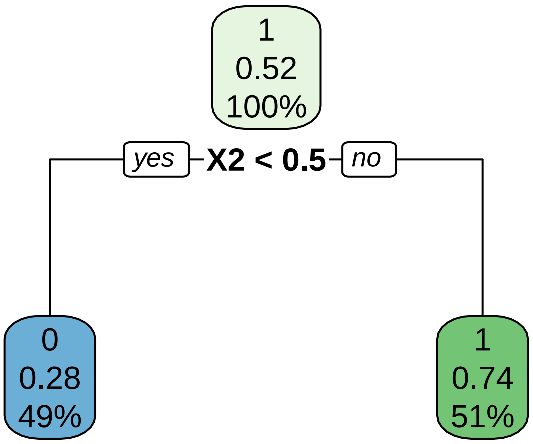
```
## Call:
## rpart(formula = Y ~ ., data = train, method = "class", parms = list(split = "gini"),
##     control = rpart.control(minsplit = 1, maxdepth = 1, maxsurrogate = 1))
##   n= 500
##
##          CP nsplit rel error    xerror       xstd
## 1 0.7261411      0 1.0000000 1.0000000 0.04636138
## 2 0.0100000      1 0.2738589 0.2738589 0.03140616
##
## Variable importance
## X1 X2
## 62 38
##
## Node number 1: 500 observations,    complexity param=0.7261411
##   predicted class=1  expected loss=0.482  P(node) =1
##     class counts:   241   259
##    probabilities: 0.482 0.518
##   left son=2 (243 obs) right son=3 (257 obs)
##   Primary splits:
##       X1 < 0.5 to the left,  improve=135.15930000, (0 missing)
##       X2 < 0.5 to the left,  improve= 52.78111000, (0 missing)
##       X3 < 0.5 to the left,  improve=  0.29823390, (0 missing)
##       X4 < 0.5 to the right, improve=  0.27751560, (0 missing)
##       X5 < 0.5 to the left,  improve=  0.05810746, (0 missing)
##   Surrogate splits:
##       X2 < 0.5 to the left,  agree=0.806, adj=0.601, (0 split)
##
```

```
## Node number 2: 243 observations
##   predicted class=0  expected loss=0.1399177  P(node) =0.486
##     class counts:   209    34
##    probabilities: 0.860 0.140
##
## Node number 3: 257 observations
##   predicted class=1  expected loss=0.1245136  P(node) =0.514
##     class counts:    32   225
##    probabilities: 0.125 0.875
```




```
## Call:
## rpart(formula = Y ~ ., data = train[, c(surrogate_split(train),
##     "Y")], method = "class", parms = list(split = "gini"), control = rpart.control(minsplit = 1,
##     maxdepth = 1, maxsurrogate = 0))
##   n= 500
##
##          CP nsplit rel error   xerror       xstd
## 1 0.439834      0  1.000000 1.074689 0.04636138
## 2 0.010000      1  0.560166 0.560166 0.04119185
##
## Variable importance
##  X2
## 100
##
## Node number 1: 500 observations,    complexity param=0.439834
##   predicted class=1  expected loss=0.482  P(node) =1
##     class counts:   241   259
##    probabilities: 0.482 0.518
##   left son=2 (246 obs) right son=3 (254 obs)
##   Primary splits:
##       X2 < 0.5 to the left,  improve=52.78111, (0 missing)
##
## Node number 2: 246 observations
##   predicted class=0  expected loss=0.2845528  P(node) =0.492
##     class counts:   176    70
##    probabilities: 0.715 0.285
##
## Node number 3: 254 observations
##   predicted class=1  expected loss=0.2559055  P(node) =0.508
##     class counts:    65   189
```

```
##    probabilities: 0.256 0.744
```



a(ii)

```
##         Overall
## X1 0.2703185498
## X2 0.1055622298
## X3 0.0005964679
## X4 0.0005550311
## X5 0.0001162149
```

```
##         Overall
## X1 0.2703185498
## X2 0.1071742298
## X3 0.0005964679
## X4 0.0005550311
## X5 0.0001162149
```

We can see the variable importance measure stays the same for all but X_2, but since X_2 is the best surrogate split, it's variable importance increases since X_2 is the best surrogate split and it's variable importance increases according to Equation(3).

a(iii)

```
## [1] 0.1
```

```
## [1] 0.27
```

The MSE of prediction based on the best split is 0.1 and based on the best surrogate split is 0.27.

b(i)

```
##   split     surrogate
##  X1  :173   NA's:1000
##  X2  :188
##  X3  : 91
##  X4  : 92
##  X5  : 74
##  NA's:382
```

```
##    split      surrogate
##   X1  :380   X2  :108
##   X2  :319   X3  :254
##   X3  : 61   X4  :248
##   X4  : 57   X5  :258
##   X5  : 51   NA's:132
##   NA's:132
```

```
##    split      surrogate
##   X1  :588   X2  :321
##   X2  :315   X3  :212
##   X3  : 25   X4  :180
##   X4  : 15   X5  :251
##   X5  : 21   NA's: 36
##   NA's: 36
```

```
##   split    surrogate
##   X1:789   X2:588
##   X2:211   X3:153
##            X4: 59
##            X5:200
```

```
##   split     surrogate
##   X1:1000   X2:1000
```

From the result of the best split and best surrogate splits, we can clearly see X_1 is the most important variable, and X_2 is the second most important. When X_1 is available, the decision stump will always split on X_1, which suggests the importance of X_1.

b(ii)

Table 1: Variable Importance

	K=1	K=2	K=3	K=4	K=5
X1	15.396471	25.7340120	37.316428	60.53472	280.8954
X2	15.861101	21.3693858	19.632084	15.68976	0.0000
X3	-1.982575	-0.5458878	-2.399697	0.00000	0.0000
X4	-1.621899	-1.1375238	1.270145	0.00000	0.0000
X5	-3.425965	-4.4428796	-3.300753	0.00000	0.0000

From the variable importance calculations, it also suggests X_1 is the most important variable and X_2 is the second most important.

Table 2: Variable Importance

	K=1	K=2	K=3	K=4	K=5
X1	15.417074	25.7906643	37.422835	60.89538	314.642
X2	15.882677	21.4037404	19.661939	15.71479	0.000
X3	-1.978897	-0.5532798	-2.401873	0.00000	0.000
X4	-1.621358	-1.1376382	1.267507	0.00000	0.000
X5	-3.427785	-4.4482939	-3.303022	0.00000	0.000

The Out-Of-Bag error decreases when K increases. Equation(2) only takes the best split into consideration, which means the second best split option does not show up on the variable importance measure. But with equation(5) and (6), each time a variable is the best surrogate split, it's also taken into consideration, which will lessen the issue of masking.

b(iii)

```
## [1] 0.224
## [1] 0.188
## [1] 0.132
## [1] 0.132
## [1] 0.132
## [1] 0.000964
## [1] 0.000264
## [1] 0.000264
## [1] 0.000264
## [1] 0.000264
```

I would say the first method correctly computed random forest's prediction error since random forest generally use majority vote decide prediction.

c(i)

Table 3: Variable Importance

	q=0.4	q=0.5	q=0.6	q=0.7	q=0.8
X1	21.1123055	29.4213333	31.8837514	36.5675412	42.9665586
X2	5.6958735	7.3640406	9.1469233	10.5816463	12.8535665
X3	0.1153367	0.1179529	0.1411444	0.1467995	0.1161901
X4	0.1224755	0.1079182	0.1008301	0.1061674	0.1379117
X5	0.0957416	0.0870237	0.0851155	0.0968633	0.0866873

The table suggests X_1 is the most important variable while X_2 is the second most important variable.

c(ii)

Table 4: Standard Deviation for Variable Importance

	X1	X2	X3	X4	X5
q=0.4	7.254624	7.033271	1.420469	1.612270	1.968850
q=0.5	7.496655	8.609039	2.161146	1.967982	2.205757
q=0.6	8.257621	9.716412	1.574322	2.326064	2.401902
q=0.7	9.103467	10.568399	1.933625	1.909720	2.592440
q=0.8	10.347065	12.151286	2.315852	2.343501	2.382830