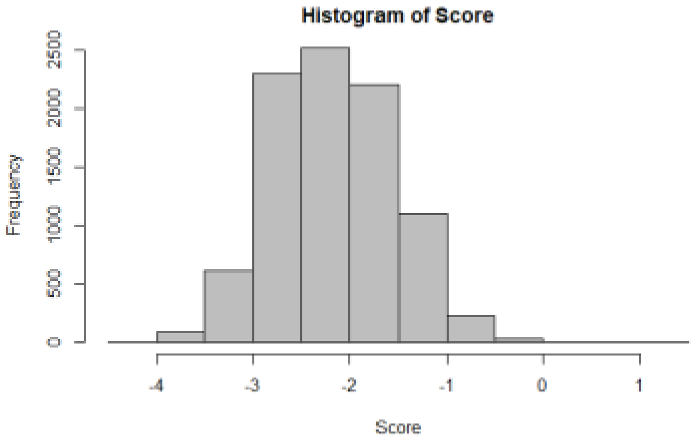
46-927 **Homework** 2

Pittsburgh
Jingyi Guo

1.

(a)

```
hist(score,xlab='Score',col='gray',main='Histogram of Score')
```

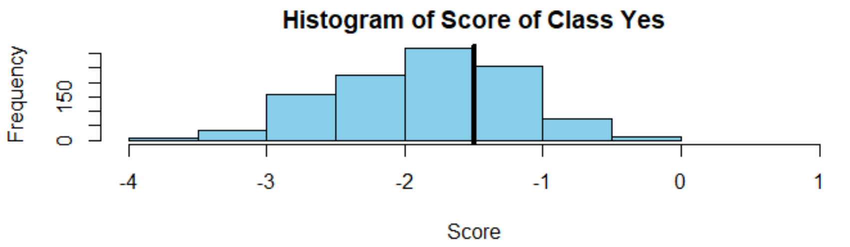


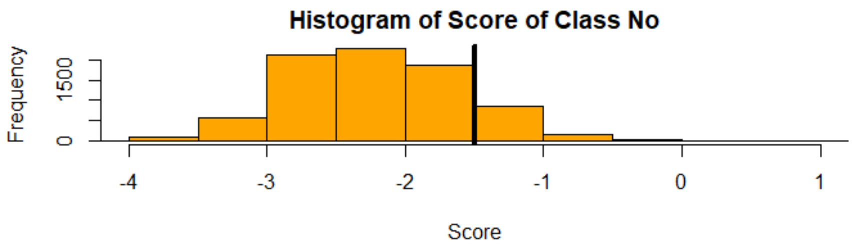
(b)

```
idx.test.yes = which(test$y=='yes')
idx.test.no = which(test$y=='no')
score.yes = score[idx.test.yes]
score.no = score[idx.test.no]
par(mfrow=c(2,1))
hist(score.yes, xlim=c(-4,1), col='skyblue', xlab='Score', main='Histogram of
Score of Class Yes')
hist(score.no, xlim=c(-4,1), col='orange', xlab='Score', main='Histogram of
Score of Class No')
```

(c)

```
par(mfrow=c(2,1))
hist(score.yes, xlim=c(-4,1), col='skyblue', xlab='Score', main='Histogram of Score of Class Yes')
abline(v=-1.5, lwd=4)
hist(score.no, xlim=c(-4,1), col='orange', xlab='Score', main='Histogram of Score of Class No')
abline(v=-1.5, lwd=4)
```





Thick black line denotes cutoff. The ratio of the area to the right of the cutoff against that of the entire histogram represents the TPR for the figure above, and FPR for the figure below.

1.(c) $TPR = \frac{TP}{TP+FN} = \frac{\int_T^\infty f_1(x)\,dx}{\int_{-\infty}^{\infty} f_1(x)\,dx} = \int_T^\infty f_1(x)\,dx$

$FPR = \frac{FP}{TN+FP} = \frac{\int_T^\infty f_0(x)\,dx}{\int_{-\infty}^{\infty} f_0(x)\,dx} = \int_T^\infty f_0(x)\,dx$

(d) $AUC = \int_{+\infty}^{-\infty} TPR(T) \cdot \frac{\partial}{\partial T} FPR(T)\,dT = \int_{\infty}^{-\infty} TPR(T) \frac{\partial}{\partial T} \int_T^\infty f_0(x)\,dx \cdot dT$

$= \int_{\infty}^{-\infty} TPR(T)\,(-f_0(T))\,dT$

$= \int_{\infty}^{-\infty} TPR(T) \cdot (-FPR'(T))\,dT = \int_{-\infty}^{\infty} TPR \cdot FPR'(T)\,dT$

where FPR is a function of x, $FPR'(x) = f_0(x)$

(e) $AUC = \int_{-\infty}^{\infty} TPR(T) \cdot FPR'(T)\,dT$

$= \int_{-\infty}^{\infty} TPR(T) \cdot f_0(T)\,dT$

$= \int_{-\infty}^{\infty} \int_T^\infty f_1(x)\,dx \cdot f_0(T)\,dT$

$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} 1_{\{x>T\}} f_1(x) f_0(T)\,dx\,dT$

$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} 1_{\{S_1>S_0\}} f_1(S_1) f_0(S_0)\,dS_1\,dS_0$

$= P(S_1 > S_0)$

2.

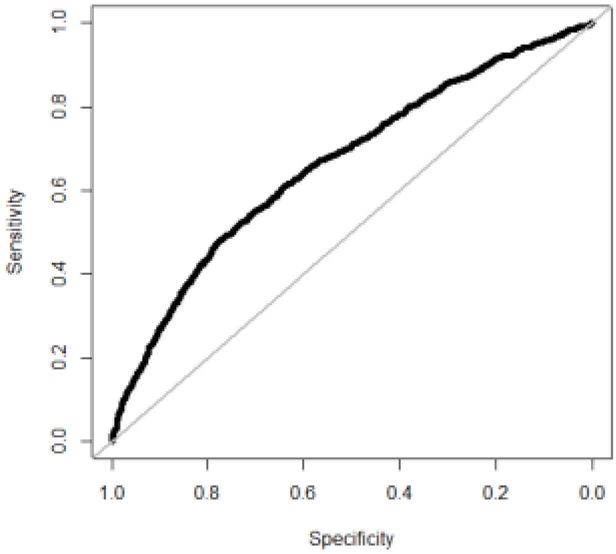
(a)

```
eval_set = function(estimate, truth, loss_FP, loss_FN){
  TP = sum((truth)&(estimate))
  FP = sum((!truth)&(estimate))
  TN = sum((!truth)&(!estimate))
  FN = sum((truth)&(!estimate))
  sens = TP / (TP+FN)
  spec = TN / (TN+FP)
  loss = loss_FP * FP + loss_FN * FN
  list(sens=sens, spec=spec, loss=loss)
}
loss_FP = 5
loss_FN = 100
estimate = guess_lm > 0.3
```

```
truth = test$y
test = eval_set(estimate, truth, loss_FP, loss_FN)
```

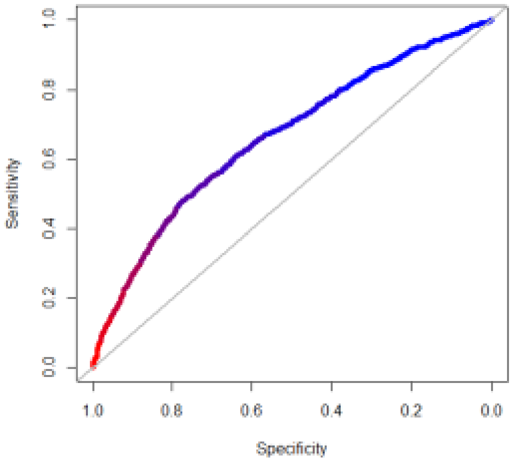
(b)

```
score_vals = sort(unique(guess_lm))
midpts = (score_vals[-1]+score_vals[-length(score_vals)])/2
output =
lapply(midpts,FUN=function(x){eval_set(guess_lm>x,truth,loss_FP,loss_FN)} )
eval_values = matrix(unlist(output), ncol=3, byrow=TRUE)
plot(eval_values[,2], eval_values[,1], xlab='Specificity', ylab='Sensitivity',
xlim =c(1.0, 0.0), pch=20)
abline(a=1, b=-1, col='gray', lwd=2)
```



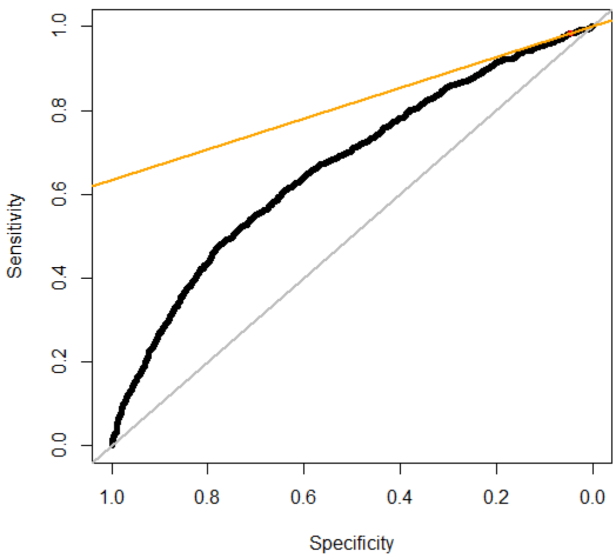
(c)

```
values = eval_values[,3]
palette = colorRampPalette(c('blue','red'))(10)
colors = palette[as.numeric(cut(values, breaks = 10))]
plot(eval_values[,2], eval_values[,1], xlab='Specificity', ylab='Sensitivity',
xlim =c(1.0, 0.0), col=colors, pch=20)
abline(a=1, b=-1, col='gray', lwd=2)
```



(d)

```r
idx.min.loss = which.min(values)
eval_values[idx.min.loss,1]
## [1] 0.9852262
eval_values[idx.min.loss,2]
## [1] 0.04485488
line.slope = -(length(idx.test.no)/length(idx.test.yes)*loss_FP/loss_FN)
line.intercpt = eval_values[idx.min.loss,1] - line.slope *
eval_values[idx.min.loss ,2]
plot(eval_values[,2], eval_values[,1], xlab='Specificity', ylab='Sensitivity',
xlim =c(1.0, 0.0), pch=20)
abline(a=1, b=-1, col='gray', lwd=2)
points(eval_values[idx.min.loss,2], eval_values[idx.min.loss,1], col='red',
pch=20)
abline(a=line.intercpt, b=line.slope, col='orange', lwd=2)
```



The sensitivity and specificity of the min loss point are 0.9852262 and 0.04485488 respectively. The orange line is tangent to the curve at minimum loss point.

2. (e) $TL(T) = FP \cdot L_{FP} + FN \cdot L_{FN}$

$= FPR \cdot N \cdot L_{FP} + FNR \cdot P \cdot L_{FN}$

$= \int_T^{\infty} f_0(x)\,dx \; N \, L_{FP} + \int_{-\infty}^{T} f_1(x)\,dx \; P \, L_{FN}.$

So $TL'(T) = -f_0(T) N L_{FP} + f_1(T) P L_{FN} = 0$

$\therefore \frac{f_1(T)}{f_0(T)} = \frac{N}{P} \frac{L_{FP}}{L_{FN}}$

3.

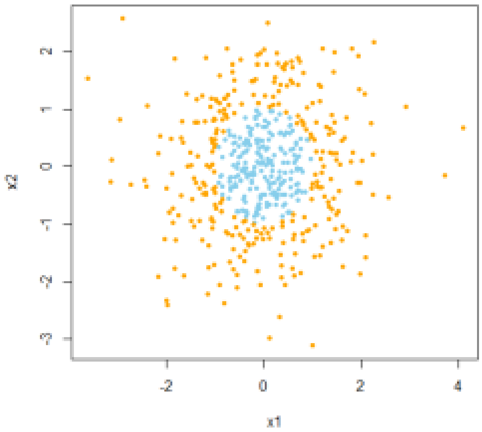
(a)

```r
source('adaboost_helpers.R')
get_circle_data = function(n){
  X = matrix(rnorm(2*n), ncol=2)
  Y = as.numeric(X[,1]^2+X[,2]^2<1)
  list(x1=X[,1], x2=X[,2], y=Y)
}
my_adaboost = function(pts, B=10){
  n = length(pts$y)
  wgts = rep(1/n, n)
  trees = vector('list', length=B)
  alphas = numeric(B)
  for(b in 1:B){
    split = find_split(pts, wgts)
    pred = predict(split$tree, type='class')
    mis = as.numeric(pred != pts$y)
    eb = sum(mis*wgts) / sum(wgts)
    ab = log((1-eb)/eb)
    trees[[b]] = split$tree
    alphas[b] = ab
    wgts = wgts * exp(ab*mis)
  }
  list(trees=trees, alphas=alphas)
}
predict_ada = function(btrees, pts){
```

```
  n = length(pts$y)
  answers = pts$y
  score = numeric(n)
  B = length(btrees$alphas)
  test_err = numeric(B)
  for(b in 1:B){
    ab = btrees$alphas[b]
    tree = btrees$trees[[b]]
    pred = as.numeric(as.character(predict(tree, newdata=pts, type='class')))*2-
1
    score = score + ab * pred
    test_err[b] = sum(as.numeric(score>0)!=answers) / n
  }
  list(score=score, predictions=as.numeric(score>0), test_err=test_err)
}
calculate_partial_dependence_x1 = function(btrees, pts){
  xlim = range(pts$x1)
  x = seq(from=xlim[1], to=xlim[2], length.out=50)
  n = length(x)
  pdep = numeric(n)
  fake = pts
  for(i in 1:n){
    fake$x1 = rep(x[i],length(fake$x2))
    pdep[i] = mean(predict_ada(btrees, fake)$predictions)
  }
  list(x=x,pdep=pdep)
}
```

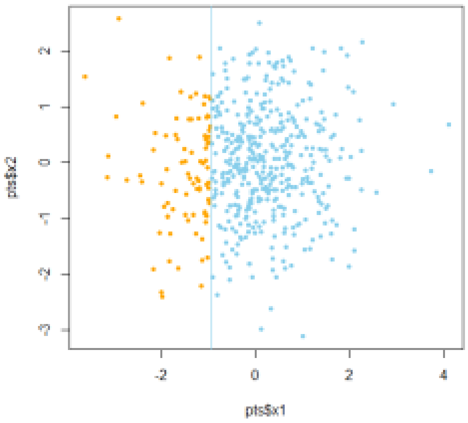
(b)

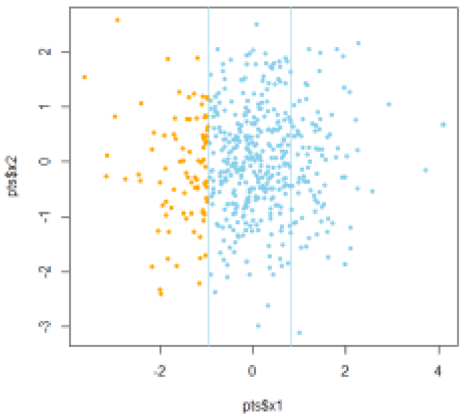
```
n = 500
train = get_circle_data(n)
test = get_circle_data(n)
train.colors = train$y + 1
plot(train$x1, train$x2, col=train.colors, xlab='x1', ylab='x2', pch=20)
```

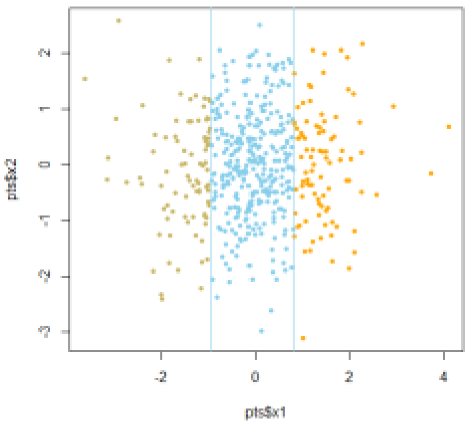


Blue spots denote class 1, orange spots denote class 0.

```
for (B in 1:3) {
  btrees = my_adaboost(train, B)
  draw_boosted_trees(btrees, train)
}
```



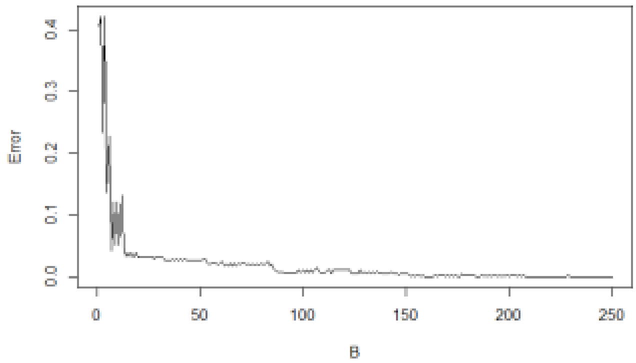




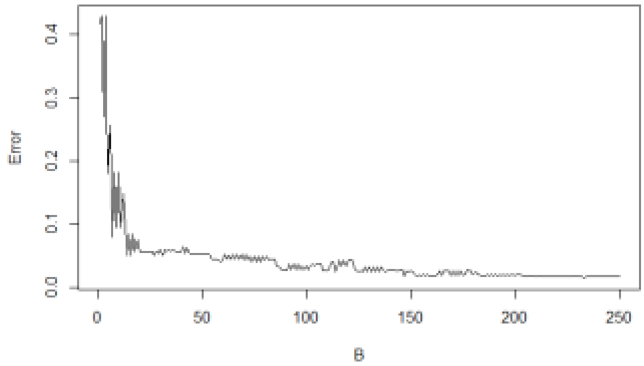
These plots demonstrate how the first 3 stumps help to classify each point.

(c)

```
B = 250
btrees = my_adaboost(train, B)
train.pred = predict_ada(btrees, train)
test.pred = predict_ada(btrees, test)
plot(1:250, train.pred$test_err, type='l', xlab='B', ylab='Error')
```

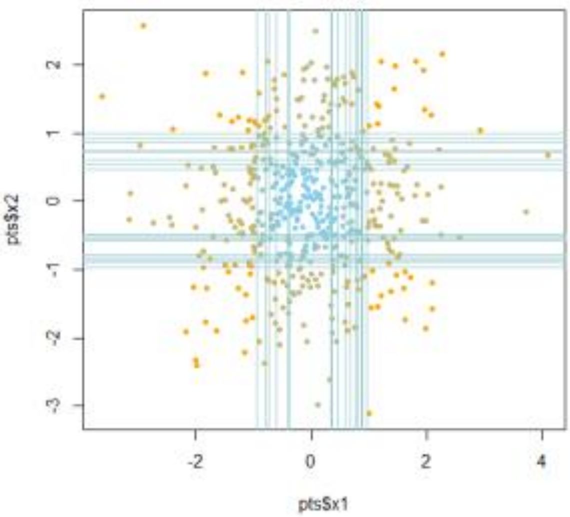



```
plot(1:250, test.pred$test_err, type='l', xlab='B', ylab='Error')
```



These 2 plots demonstrate the error rate of training set and test set as the number of trees increases from 0 to 250. We can see that the AdaBoost algorithm produces fairly good results when B>20.

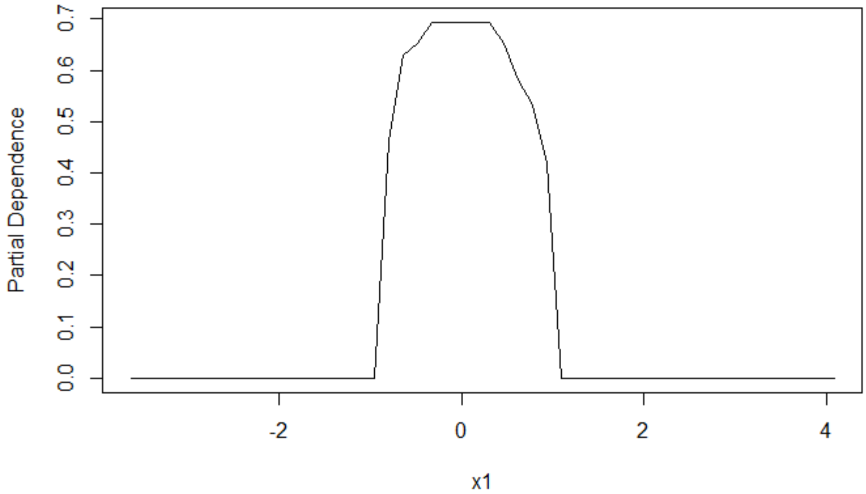
```
draw_boosted_trees(btrees, train)
```



When B is large, almost all the points within the unit circle are correctly identified and labelled.

(d)

```
pdep = calculate_partial_dependence_x1(btrees, train)
plot(pdep$x, pdep$pdep, type='l', xlab='x1', ylab='Partial Dependence')
```



The points within [-1,1] are very likely to be positive samples.