Homework Assignment 2

Matthew Xu

8 November 2020

```r
# knit from directory
setwd("/Users/MatthewXu/Downloads/")
library(readr)

knitr::opts_knit$set(root.dir = "/Users/MatthewXu/Downloads/")
```

```r
library(tidyverse)
library(tree)
library(plyr)
library(class)
library(rpart)
library(maptree)
library(ROCR)
```

```r
#read in spambase.tab and standardize each numerical attribute in the dataset
#standardized column have 0 mean and unit variance
spam <- read_table2("spambase.tab", guess_max=2000)
spam <- spam %>%
mutate(y = factor(y, levels=c(0,1), labels=c("good", "spam"))) %>% # label as factors
mutate_at(.vars=vars(-y), .funs=scale) # scale others

#calculate misclassification error rate
calc_error_rate <- function(predicted.value, true.value){
return(mean(true.value!=predicted.value))
}

#matrix called records to keep track of error rates
records = matrix(NA, nrow=3, ncol=2)
colnames(records) <- c("train.error","test.error")
rownames(records) <- c("knn","tree","logistic")

#split randomely data into training and test data set
set.seed(1)
test.indices = sample(1:nrow(spam), 1000)
```

```r
spam.train=spam[-test.indices,]
spam.test=spam[test.indices,]

#10 fold cross validation
#ensuredata partioning is consistent define folds with fold assignments for each observation i
nfold = 10
set.seed(1)
folds = seq.int(nrow(spam.train)) %>% ## sequential obs ids
cut(breaks = nfold, labels=FALSE) %>% ## sequential fold ids
sample ## random fold ids
```

1. "PROBLEM NUMBER ONE"

```r
set.seed(1)

kvec = c(1, seq(10, 50, length.out=5))

do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){
train = (folddef!=chunkid)
Xtr = Xdat[train,]
Ytr = Ydat[train]
Xvl = Xdat[!train,]
Yvl = Ydat[!train]

## get classifications for current training chunks
predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)
## get classifications for current test chunk
predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)

data.frame(train.error = calc_error_rate(predYtr, Ytr),
val.error = calc_error_rate(predYvl, Yvl))
}
set.seed(1)
errors = NULL

for(k in kvec){
  #cross validation using do.chunk
  #x data without y labels with y
  tmp = ldply(1:nfold, do.chunk, folddef=folds, Xdat = dplyr::select(spam.train, -y),
              Ydat=spam.train$y, k = k) %>%
    summarise_all(funs(mean)) %>% #find mean of all training/test errors
  mutate(neighbors = k) #name of k to neighbors

  #bind to empty dataframe for each iteration of fold
  errors = rbind(errors, tmp)
}

#find smallest val.error
min(errors$val.error)
```

```
## [1] 0.09664204
```

```r
errors
```

```
##    train.error  val.error neighbors
## 1 0.0003394205 0.10358033         1
## 2 0.0788670906 0.09664204        10
## 3 0.0924125879 0.10219606        20
## 4 0.1008669973 0.10913820        30
## 5 0.1069764133 0.11246845        40
## 6 0.1113270938 0.11552478        50
```

Therefore, the minmum estimated val.error of 0.1005255 is assigned to number of neighbors of 10, k = 10 leads to smallest estimated test error

2. "PROBLEM NUMBER TWO"

```r
# from #1 best neighbor number is 10
best.kfold = 10
set.seed(1)
cl = spam.train$y

# Train
pred.YTrain = knn(train = spam.train[, -58], test = spam.train[, -58], cl = cl, k = best.kfold
errorrate_train <- calc_error_rate(pred.YTrain, spam.train$y)
errorrate_train
```

```
## [1] 0.07803388
```

```r
# Test
pred.YTest = knn(train = spam.train[, -58], test = spam.test[, -58], cl = cl, k = best.kfold)
errorrate_test <- calc_error_rate(pred.YTest, spam.test$y)
errorrate_test
```

```
## [1] 0.102
```

```
# fill in first row of records with knn fit
records[1, ] <- c(errorrate_train, errorrate_test)
records
```

```
##          train.error test.error
## knn       0.07803388      0.102
## tree              NA         NA
## logistic          NA         NA
```

3. "PROBLEM NUMBER THREE"

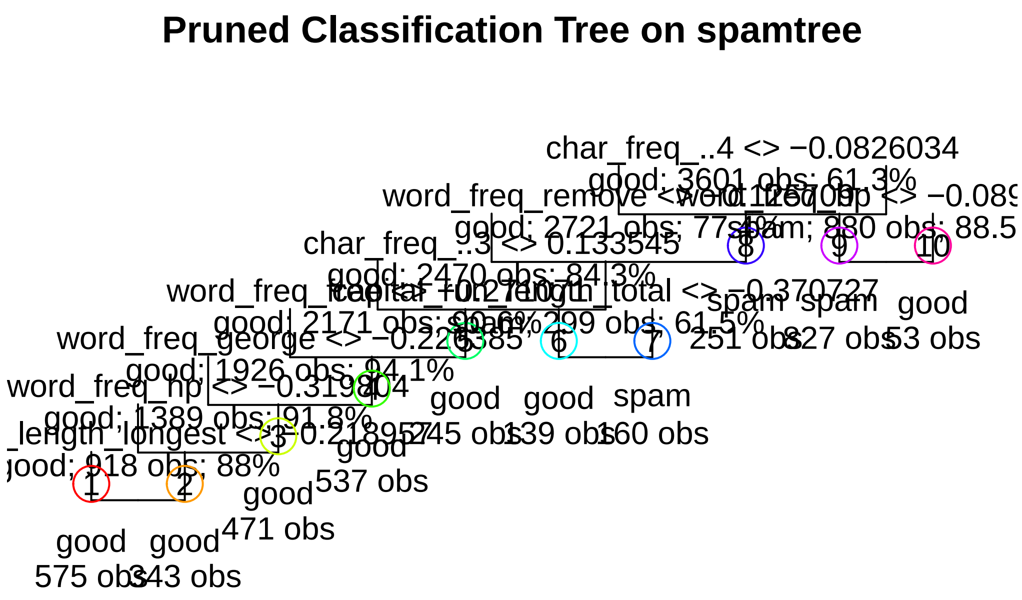
```
# creating decsion tree with control paramaters control first paramater is number
# of observations
spamtree <- tree(y ~ ., spam.train, control = tree.control(nrow(spam.train), minsize = 5,
    mindev = 1e-05))
summary(spamtree)
```

```
##
## Classification tree:
## tree(formula = y ~ ., data = spam.train, control = tree.control(nrow(spam.train),
##     minsize = 5, mindev = 1e-05))
## Variables actually used in tree construction:
##  [1] "char_freq_..4"              "word_freq_remove"
##  [3] "char_freq_..3"              "word_freq_free"
##  [5] "word_freq_george"           "word_freq_hp"
##  [7] "capital_run_length_longest" "word_freq_receive"
##  [9] "word_freq_credit"           "capital_run_length_average"
## [11] "word_freq_your"             "word_freq_mail"
## [13] "word_freq_re"               "word_freq_our"
## [15] "word_freq_you"              "capital_run_length_total"
## [17] "word_freq_make"             "word_freq_all"
## [19] "word_freq_internet"         "word_freq_email"
## [21] "word_freq_project"          "word_freq_money"
## [23] "word_freq_1999"             "word_freq_will"
## [25] "char_freq_..1"              "word_freq_order"
## [27] "char_freq_."                "word_freq_data"
## [29] "word_freq_over"             "word_freq_meeting"
## [31] "word_freq_650"              "word_freq_edu"
## [33] "word_freq_address"          "word_freq_business"
## Number of terminal nodes:  149
## Residual mean deviance:  0.04568 = 157.7 / 3452
## Misclassification error rate: 0.01361 = 49 / 3601
```

There appear to be 184 leaf (terminal) nodes. The misclassification error rate is 48/3601 meaning there are 48 misclassified training observation.

4. “PROBLEM NUMBER FOUR”

```r
# pruning tree to leaf node 10 from piazza, take out method = 'misclass' since
# least effects on overall misclassification
prune = prune.tree(spamtree, best = 10)

# depict tree
draw.tree(prune, nodeinfo = TRUE)
title("Pruned Classification Tree on spamtree")
```

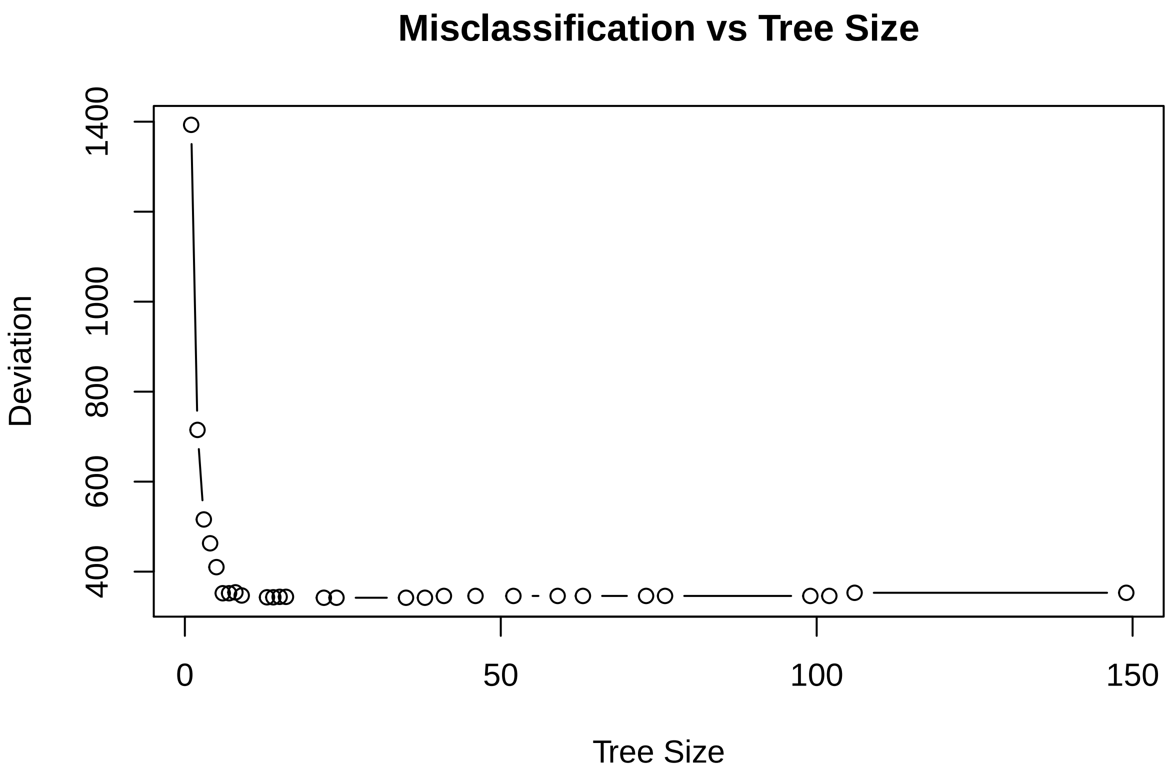


# 5. “PROBLEM NUMBER FIVE”

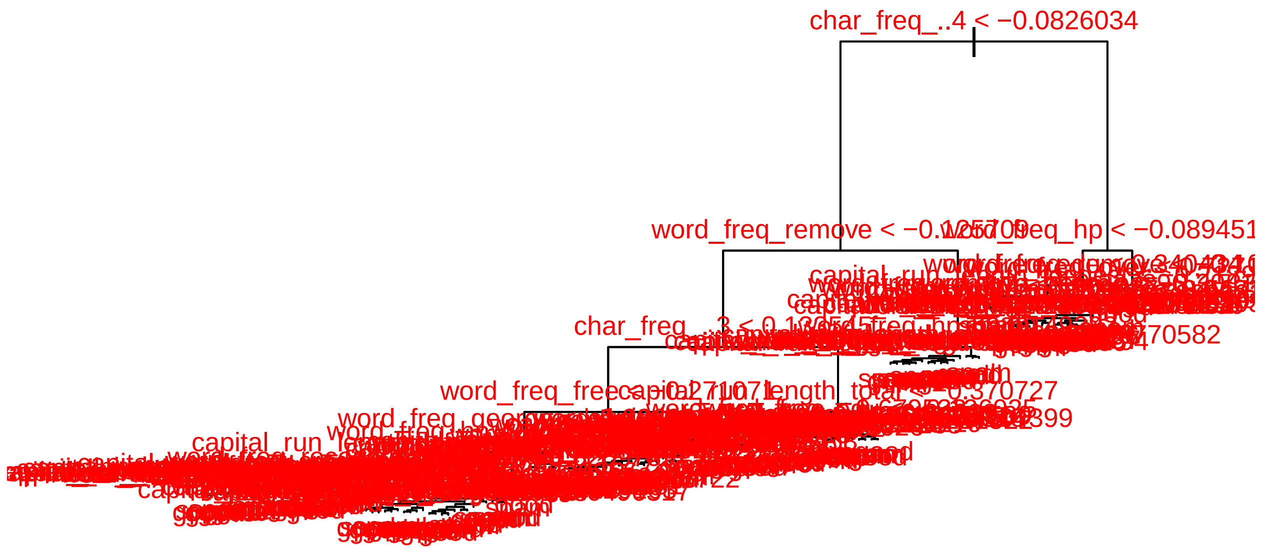
```r
set.seed(1)

# K-Fold cross validation rand is number of cases, the fold partioning
cv = cv.tree(spamtree, rand = folds, K = 10, method = "misclass")

# find best tree size there are identical deviations for different sizes find
# last location where the deviations are identical, then size index - 1
best.size.cv = cv$size[which.min(cv$dev == min(cv$dev)) - 1]
best.size.cv
```

```
## integer(0)
```

```r
# plot size vs dev
plot(cv$size, cv$dev, type = "b", xlab = "Tree Size", ylab = "Deviation", main = "Misclassifica

# show best.size.cv in the plot
abline(v = best.size.cv, col = "blue")
```



```r
pt.prune = prune.misclass(spamtree, best = best.size.cv)
# Plot pruned tree
plot(pt.prune)
text(pt.prune, pretty = 0, col = "red", cex = 0.8)
```


char_freq_..4 < −0.0826034
word_freq_remove < −0.125709
word_freq_hp < −0.089451
char_freq_..3 < 0.1365
word_freq_free
capital_run_length_total < −0.370727
word_freq_george
capital_run

6. “PROBLEM NUMBER SIX”

```r
# pruned tree with best size from cv
spamtree.pruned = prune.tree(spamtree, best = best.size.cv)

# train error
pruned.predict.train <- predict(spamtree.pruned, spam.train[, -58], type = "class")
tree.train <- calc_error_rate(pruned.predict.train, spam.train$y)
tree.train
```

```
## [1] 0.01610664
```

```r
# test error
pruned.predict.test <- predict(spamtree.pruned, spam.test[, -58], type = "class")
tree.test <- calc_error_rate(pruned.predict.test, spam.test$y)
tree.test
```

```
## [1] 0.09
```

```r
# put errors in row 2
records[2, ] <- c(tree.test, tree.train)
records
```

```
##          train.error test.error
## knn       0.07803388 0.10200000
## tree      0.09000000 0.01610664
## logistic          NA         NA
```

7. “PROBLEM NUMBER SEVEN”

Handwritten, shown below attached to this .rmd pdf

8. “PROBLEM NUMBER EIGHT”

```r
# fitting logitisc regression to data
glm.fit <- glm(spam$y ~ ., data = spam, family = binomial)

# Specify type='response' to get the estimated probabilities predcting using the
# logistic regression with x values of train/test
prob.training <- predict(glm.fit, spam.train[, -58], type = "response")
prob.testing <- predict(glm.fit, spam.test[, -58], type = "response")
```

```r
# mutatate each respective train/test sets with prediction factor using threshold
# mutated to values being 'good', 'spam'
spam.train = spam.train %>% mutate(pred.trainSPAM = as.factor(ifelse(prob.training <=
    0.5, "good", "spam")))
spam.test = spam.test %>% mutate(pred.testSPAM = as.factor(ifelse(prob.testing <=
    0.5, "good", "spam")))

# confusion matrix of train and test
table(pred = spam.train$pred.trainSPAM, true = spam.train$y)
```

```
##       true
## pred   good spam
##   good 2108  144
##   spam  100 1249
```

```r
table(pred = spam.test$pred.testSPAM, true = spam.test$y)
```

```
##       true
## pred   good spam
##   good  558   50
##   spam   22  370
```

```r
# train error
log.Train_Error <- calc_error_rate(spam.train$pred.trainSPAM, spam.train$y)
log.Train_Error
```

```
## [1] 0.06775896
```

```r
# test error
log.Test_Error <- calc_error_rate(spam.test$pred.testSPAM, spam.test$y)
log.Test_Error
```

```
## [1] 0.072
```

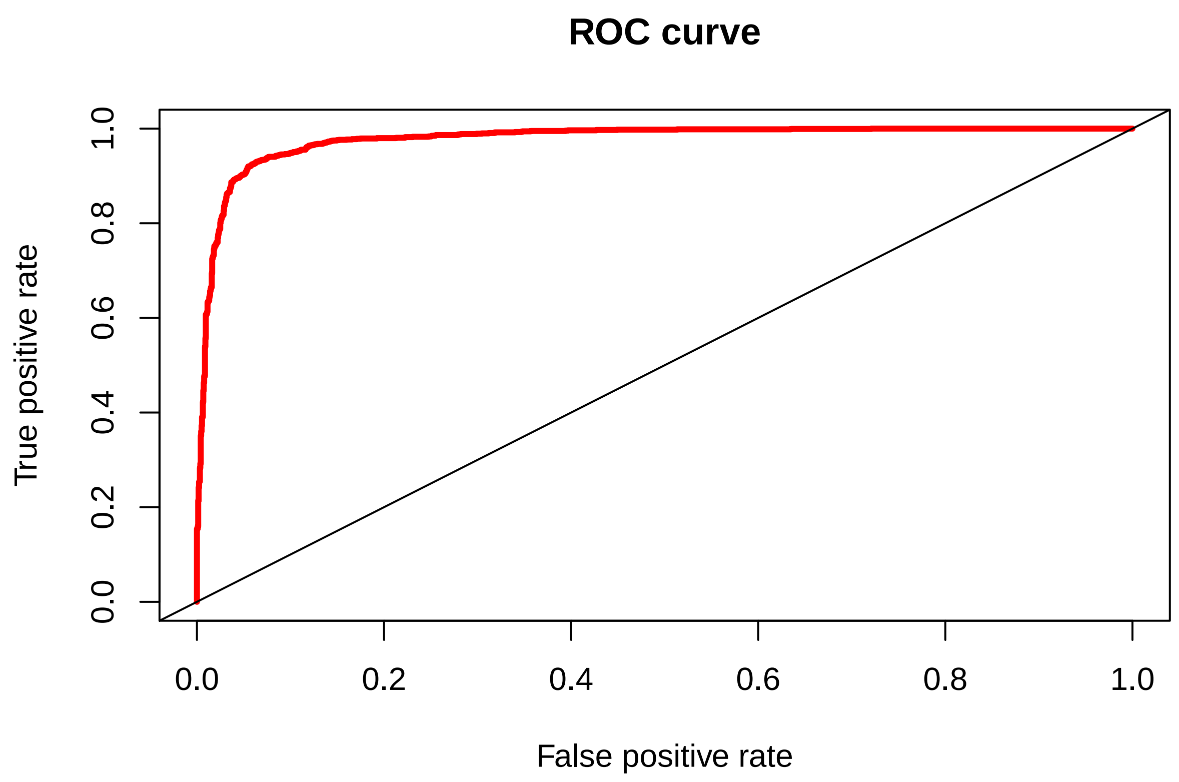
```r
# insert test/train error to row 3
records[3, ] <- c(log.Test_Error, log.Train_Error)
records
```

```
##          train.error test.error
## knn       0.07803388 0.10200000
## tree      0.09000000 0.01610664
## logistic  0.07200000 0.06775896
```

The method with the lowest test error is the decsion tree method, having the lowest at 0.05554013. Next is logisitc regression at 0.06886976 and the worst being knn with the largest error of 0.09.

9. "PROBLEM NUMBER NINE"

```
# prediction used to standardize data of proabailbites to labels
pred = prediction(prob.training, spam.train$y)
# measing preformance
perf = performance(pred, measure = "tpr", x.measure = "fpr")
# plot ROC curve
plot(perf, col = 2, lwd = 3, main = "ROC curve")
abline(0, 1)
```



```
# Calculate AUC
auc = performance(pred, "auc")@y.values
auc
```

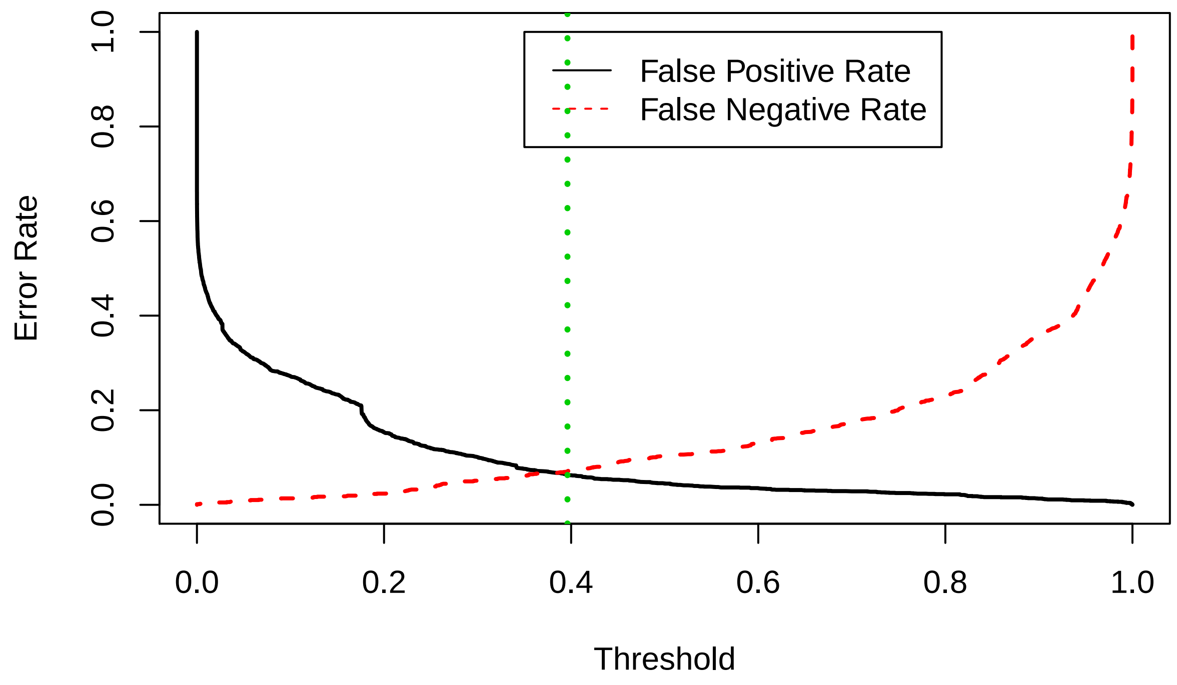
```
## [[1]]
## [1] 0.9770243
```

```
# FPR
fpr = performance(pred, "fpr")@y.values[[1]]
cutoff = performance(pred, "fpr")@x.values[[1]]
# FNR
fnr = performance(pred, "fnr")@y.values[[1]]

# find rates for threshold
```

```r
rate = as.data.frame(cbind(Cutoff = cutoff, FPR = fpr, FNR = fnr))
rate$distance = sqrt((rate[, 2])^2 + (rate[, 3])^2)

# find best threshold
index = which.min(rate$distance)
best = rate$Cutoff[index]
best
```

```
## [1] 0.3960961
```

```r
# Plot
matplot(cutoff, cbind(fpr, fnr), type = "l", lwd = 2, xlab = "Threshold", ylab = "Error Rate")
legend(0.35, 1, legend = c("False Positive Rate", "False Negative Rate"), col = c(1,
    2), lty = c(1, 2))  # Add the best value

abline(v = best, col = 3, lty = 3, lwd = 3)
```



By looking at the ROC curve and the AUC values, due to the high AUC value of 0.9761 and the curve of the graph, true positive rate is contiounly converging to 1, meaning that postive predictions are in reality correctly predicted positive. In this case, we are much more concerne about false positives, as false positives indicate that emails flagged spam are actually important emails not spam. Low true positives mean that there is a lower rate of spammed emails, which is a good thing. For logisitc regression, the model can be even better by using the calculated best threshold at 0.3961 instead of 0.5.

7.) a.) $p(z) = \frac{e^z}{1+e^z}$

$p(1+e^z) = e^z$

$p + p(e^z) = e^z$

$p = e^z - pe^z$

$p = e^z(1-p)$

$e^z = \frac{p}{1-p}$

$z = \ln(\frac{p}{1-p})$ taking ln() of both sides

$z(p) = \ln(\frac{p}{1-p})$

Therefore, the inverse of a logistic function is the logit function

7.) b.) $z = B_0 + B_1 x_1$ $p = \text{logistic}(z)$

$z = \ln(\frac{p}{1-p})$ where $\frac{p}{1-p}$ is the odds

if x_1 is instead $x_1 + 2$

$$p = \frac{e^{B_0 + B_1(x_1+2)}}{1 + e^{B_0 + B_1(x_1+2)}}$$

$$p = \frac{e^{B_0 + B_1 x_1 + 2B_1}}{1 + e^{B_0 + B_1 x_1 + 2B_1}}$$

This is the same as logistic $(z + 2B_1)$

comparing this to the logit function

$z = \ln\left(\frac{p}{1-p}\right)$

if z is $z + 2B_1$

$z + 2B_1 = \ln\left(\frac{p}{1-p}\right)$ ln both sides

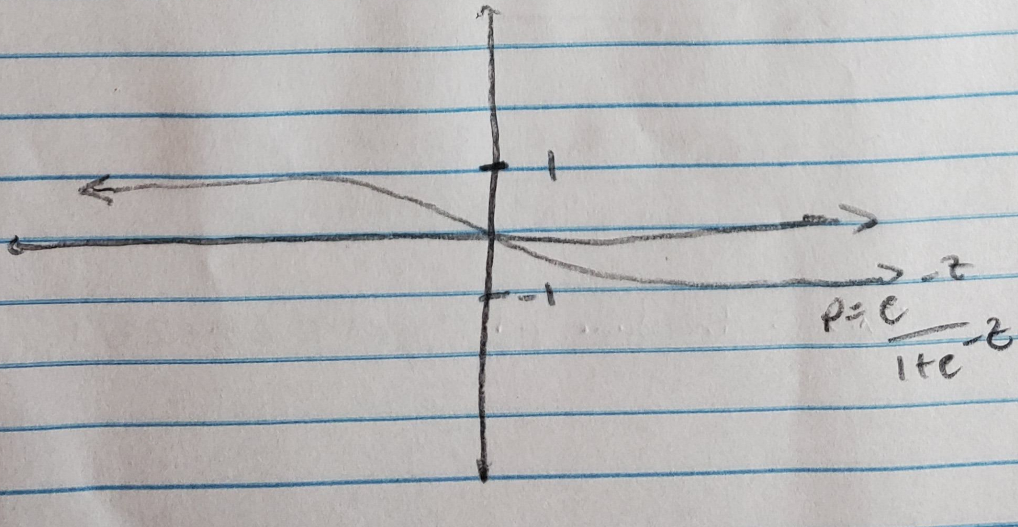
$\frac{p}{1-p} = e^{z+2B_1}$

$= e^{z} \cdot e^{2B_1}$

Therefore the odds changes to a factor of the constant e^{2B_1}

If B_1 is negative, meaning as $x_1 \to \infty$, $z \to -\infty$, because $B_1 x_1$ is becoming more negative. If $z \to -\infty$, then p approaches 0, as $p = \frac{e^z}{1+e^z}$.

If $x_1 \to -\infty$, $z \to \infty$, because $B_1 x_1$ is becoming more positive. If $z \to \infty$, then p approaches 1 as $p = \frac{e^z}{1+e^z}$


z
$p = \frac{e^{-z}}{1+e^{-z}}$