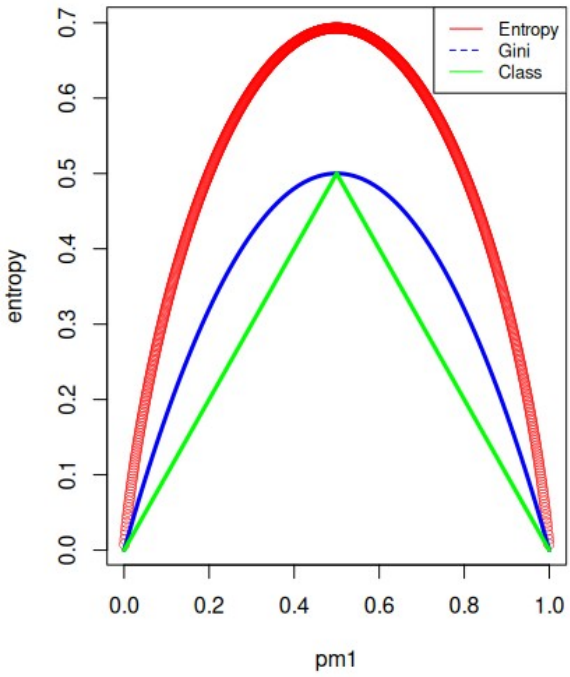
Question 3

I created a sequence from 0 to 2 stepping by .001. I defined pm1 by this sequence, and pm2 by 1 – pm1. I then created 3 formulas for each of the types of errors, and then plotted them. I got:



Question 9

a) I set the seed, and split my data into training and test

b) I created a tree, and printed the summary from the book's code. The training error rate is displayed at the bottom, of .1588. There are 9 terminal nodes in total for this tree

```
Classification tree:
tree(formula = Purchase ~ ., data = train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  9
Residual mean deviance:  0.7432 = 587.8 / 791
Misclassification error rate: 0.1588 = 127 / 800
```

c) Here is the output: Following the path to nodes 8 and 9, tell us that the first split was LoyalCH at .5036. If the value is below .5036, then the next split is if the value is below .2809. If this is then true, the third split is if the value is < .03564, then the result is MM, and if not then the result is still MM. For any value of LoyalCH less than .28, the prediction is MM.

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 800 1073.00 CH ( 0.60625 0.39375 )
   2) LoyalCH < 0.5036 365  441.60 MM ( 0.29315 0.70685 )
     4) LoyalCH < 0.280875 177  140.50 MM ( 0.13559 0.86441 )
       8) LoyalCH < 0.0356415 59   10.14 MM ( 0.01695 0.98305 ) *
       9) LoyalCH > 0.0356415 118  116.40 MM ( 0.19492 0.80508 ) *
     5) LoyalCH > 0.280875 188  258.00 MM ( 0.44149 0.55851 )
      10) PriceDiff < 0.05 79   84.79 MM ( 0.22785 0.77215 )
        20) SpecialCH < 0.5 64   51.98 MM ( 0.14062 0.85938 ) *
        21) SpecialCH > 0.5 15   20.19 CH ( 0.60000 0.40000 ) *
      11) PriceDiff > 0.05 109  147.00 CH ( 0.59633 0.40367 ) *
   3) LoyalCH > 0.5036 435  337.90 CH ( 0.86897 0.13103 )
     6) LoyalCH < 0.764572 174  201.00 CH ( 0.73563 0.26437 )
      12) ListPriceDiff < 0.235 72   99.81 MM ( 0.50000 0.50000 )
        24) PctDiscMM < 0.196196 55   73.14 CH ( 0.61818 0.38182 ) *
        25) PctDiscMM > 0.196196 17   12.32 MM ( 0.11765 0.88235 ) *
      13) ListPriceDiff > 0.235 102   65.43 CH ( 0.90196 0.09804 ) *
     7) LoyalCH > 0.764572 261   91.20 CH ( 0.95785 0.04215 ) *
```

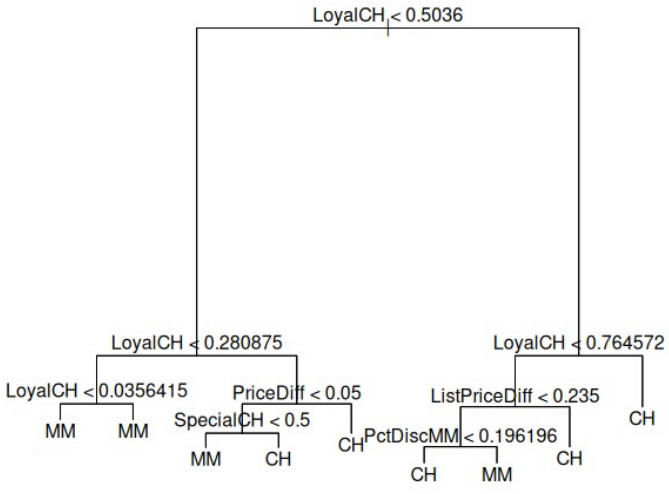
d) Pretty tree!

e) Here is my confusion matrix. The success rate is .829, meaning the test error rate is .171

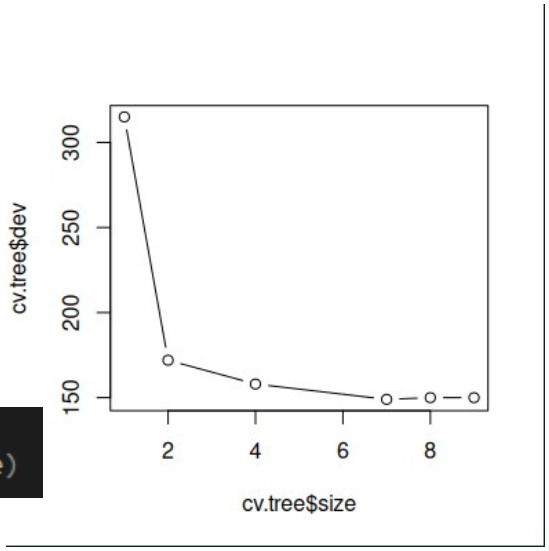
```
tree.pred  CH  MM
       CH 160  38
       MM   8  64
[1] "(160 + 64) / 270"
[1] 0.8296296
```

f) I used the code from the book to do so

```
cv.tree = cv.tree(tree, FUN = prune.misclass)
print(cv.tree)
min.dev = which.min(cv.tree$dev)
print("Which lowest dev?")
print(min.dev)
print("Which number of terminal nodes?")
best.size = cv.tree$size[min.dev]
print(best.size)
```



g) graph shown on right



h) The tree with 7 nodes corresponded to the lowest CV error rate

I) I pruned the tree

```
# i)
prune.tree = prune.misclass(tree, best = best.size)
```

j) The train error rate for the pruned tree was .1625, which is better than that of the regular tree

```
Classification tree:
snip.tree(tree = tree, nodes = c(4L, 10L))
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7748 = 614.4 / 793
Misclassification error rate: 0.1625 = 130 / 800
```

k) The test error rate for the pruned tree was better than the regular tree

```
tree.pred  CH  MM
       CH 160  38
       MM   8  64
[1] "(8 + 38) / 270"
[1] 0.1703704
[1] "----Pruned Tree-----"

tree.pred_prune  CH  MM
             CH 160  36
             MM   8  66
[1] "(8 + 36) / 270"
[1] 0.162963
```