Confusion Matrix and Classification

Akhil Gupta

February 16, 2017

#install.packages("ROCR")  
library(ROCR)

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

#  
  
  
getwd()

## [1] "D:/Sem2/1. self/dataMininingPredictiveAnalysis/HW/3. HW3"

setwd("D:/Sem2/1. self/dataMininingPredictiveAnalysis/HW/3. HW3")  
getwd()

## [1] "D:/Sem2/1. self/dataMininingPredictiveAnalysis/HW/3. HW3"

#1a  
pref<-read.csv('VoterPref.csv')  
attach(pref)  
  
PREFERENCE <- factor(PREFERENCE,levels=c("For","Against"))  
L\_PREF <- (as.numeric(PREFERENCE)-1)  
  
pref<-cbind(pref,L\_PREF)  
  
  
#1b  
set.seed(123457)  
  
#1c  
train\_ind<-sample(seq\_len(nrow(pref)),size= .7\*nrow(pref))  
train<-pref[train\_ind, ]  
test<-pref[-train\_ind, ]  
nrow(train)

## [1] 700

nrow(test)

## [1] 300

#2a  
fit\_logistic <- glm(as.numeric(L\_PREF)~AGE+INCOME+factor(GENDER), family = "binomial", data = train)  
summary(fit\_logistic)

##   
## Call:  
## glm(formula = as.numeric(L\_PREF) ~ AGE + INCOME + factor(GENDER),   
## family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.23799 -0.38579 -0.13440 -0.02922 2.81772   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.13300 0.76992 0.173 0.863   
## AGE 0.23953 0.02462 9.729 <2e-16 \*\*\*  
## INCOME -0.13184 0.01268 -10.398 <2e-16 \*\*\*  
## factor(GENDER)M -0.53005 0.27957 -1.896 0.058 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 680.71 on 699 degrees of freedom  
## Residual deviance: 340.35 on 696 degrees of freedom  
## AIC: 348.35  
##   
## Number of Fisher Scoring iterations: 7

predicted\_train\_logistic <- predict(fit\_logistic, newdata = train, type = "response")  
  
tlogistic\_train <- ifelse(predicted\_train\_logistic > 0.5,1,0)  
  
confusion\_m\_logistic\_train<-table(as.numeric(train$L\_PREF),tlogistic\_train)  
confusion\_m\_logistic\_train

## tlogistic\_train  
## 0 1  
## 0 545 22  
## 1 47 86

confusion\_m\_logistic\_train\_probability<-confusion\_m\_logistic\_train/sum(confusion\_m\_logistic\_train)  
confusion\_m\_logistic\_train\_probability

## tlogistic\_train  
## 0 1  
## 0 0.77857143 0.03142857  
## 1 0.06714286 0.12285714

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#  
predicted\_test\_logistic <- predict(fit\_logistic, newdata = test, type = "response")  
  
tlogistic\_test <- ifelse(predicted\_test\_logistic > 0.5,1,0)  
  
confusion\_m\_logistic\_test<-table(as.numeric(test$L\_PREF),tlogistic\_test)  
confusion\_m\_logistic\_test

## tlogistic\_test  
## 0 1  
## 0 225 17  
## 1 24 34

confusion\_m\_logistic\_test\_probability<-confusion\_m\_logistic\_test/sum(confusion\_m\_logistic\_test)  
confusion\_m\_logistic\_test\_probability

## tlogistic\_test  
## 0 1  
## 0 0.75000000 0.05666667  
## 1 0.08000000 0.11333333

#2b. Compute the sensitivity, specificity, accuracy, error rate, PPV, NPV for training data set  
  
accuracy\_logistic\_train<- ((confusion\_m\_logistic\_train[1,1]+confusion\_m\_logistic\_train[2,2])/(confusion\_m\_logistic\_train[1,1]+confusion\_m\_logistic\_train[2,1]+confusion\_m\_logistic\_train[1,2]+confusion\_m\_logistic\_train[2,2]))  
accuracy\_logistic\_train

## [1] 0.9014286

senstivity\_logistic\_train<-((confusion\_m\_logistic\_train[2,2])/(confusion\_m\_logistic\_train[2,1]+confusion\_m\_logistic\_train[2,2]))  
senstivity\_logistic\_train

## [1] 0.6466165

specificity\_logistic\_train<-((confusion\_m\_logistic\_train[1,1])/(confusion\_m\_logistic\_train[1,1]+confusion\_m\_logistic\_train[1,2]))  
specificity\_logistic\_train

## [1] 0.9611993

errorRate\_logistic\_train<-((confusion\_m\_logistic\_train[1,2]+confusion\_m\_logistic\_train[2,1])/(confusion\_m\_logistic\_train[1,1]+confusion\_m\_logistic\_train[2,1]+confusion\_m\_logistic\_train[1,2]+confusion\_m\_logistic\_train[2,2]))  
errorRate\_logistic\_train

## [1] 0.09857143

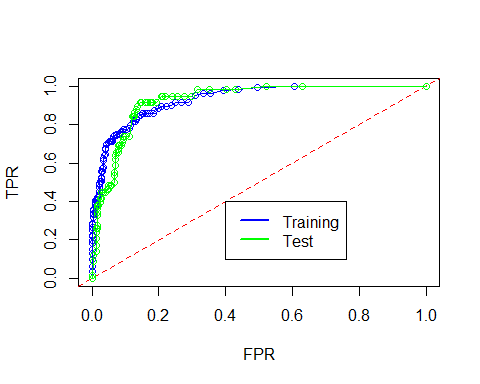
ppv\_logistic\_train<-((confusion\_m\_logistic\_train[2,2])/(confusion\_m\_logistic\_train[2,2]+confusion\_m\_logistic\_train[1,2]))  
ppv\_logistic\_train

## [1] 0.7962963

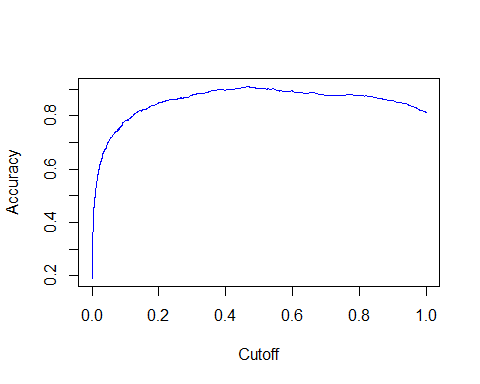
npv\_logistic\_train<-((confusion\_m\_logistic\_train[1,1])/(confusion\_m\_logistic\_train[1,1]+confusion\_m\_logistic\_train[2,1]))  
npv\_logistic\_train

## [1] 0.9206081

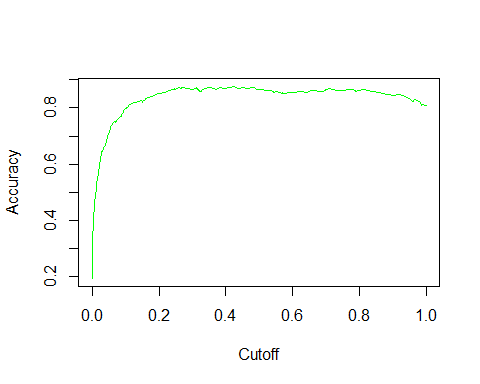
#2c. Plot the ROC curves for both the training and test data sets on the same graph (distinguishing with different colors). What can you infer from a scrutiny of this graph?  
  
cutoff <- seq(0, 1, length = 100)  
fpr\_train <- numeric(100)  
tpr\_train <- numeric(100)  
  
roc\_table\_train <- data.frame(Cutoff = cutoff, FPR = fpr\_train,TPR = tpr\_train)  
Actual\_train <- train$PREFERENCE  
#Actual\_train  
for (i in 1:100) {  
 roc\_table\_train$FPR[i] <- sum(predicted\_train\_logistic > cutoff[i] & Actual\_train == "For")/sum(Actual\_train == "For")  
 roc\_table\_train$TPR[i] <- sum(predicted\_train\_logistic > cutoff[i] & Actual\_train == "Against")/sum(Actual\_train == "Against")  
}  
  
plot(TPR ~ FPR, data = roc\_table\_train, type = "o",xlab="FPR",ylab="TPR",col="blue")  
abline(a = 0, b = 1, lty = 2,col="red")  
  
cutoff <- seq(0, 1, length = 100)  
FPR\_test <- numeric(100)  
TPR\_test <- numeric(100)  
Actual\_test <- test$PREFERENCE  
roc\_table\_test <- data.frame(Cutoff = cutoff, FPR = FPR\_test,TPR = TPR\_test)  
  
for (i in 1:100) {  
 roc\_table\_test$FPR[i] <- sum(predicted\_test\_logistic > cutoff[i] & Actual\_test == "For")/sum(Actual\_test == "For")  
 roc\_table\_test$TPR[i] <- sum(predicted\_test\_logistic > cutoff[i] & Actual\_test == "Against")/sum(Actual\_test == "Against")  
}  
lines(TPR~FPR,data = roc\_table\_test, type="o",col="green")  
legend(0.4,0.4,c("Training", "Test"),lty=c(1,1), lwd=c(2.5,2.5), col=c("blue","green"))



#2d.  
pred1<-prediction(predicted\_train\_logistic, train$L\_PREF)  
perf\_train <- performance(pred1, "acc")  
plot( perf\_train , show.spread.at=seq(0, 1, by=0.1), col="blue")



pred2<-prediction(predicted\_test\_logistic, test$L\_PREF)  
perf\_test <- performance( pred2, "acc")  
plot( perf\_test , show.spread.at=seq(0, 1, by=0.1), col="green")



#2e  
max\_accuracy\_train <- max(perf\_train@y.values[[1]])  
max\_accuracy\_train

## [1] 0.91

Cutoff\_train <- perf\_train@x.values[[1]][which.max(perf\_train@y.values[[1]])]  
Cutoff\_train

## 766   
## 0.4625541

#2f.  
flag\_test<-ifelse(predicted\_test\_logistic>0.4212197,1,0)  
flag\_test\_table<-table(as.numeric(test$L\_PREF),flag\_test)  
#flag\_test\_table  
accuracy\_test<-(flag\_test\_table[1,1]+flag\_test\_table[2,2])/(flag\_test\_table[1,1]+flag\_test\_table[2,2]+flag\_test\_table[1,2]+flag\_test\_table[2,1])  
accuracy\_test

## [1] 0.8766667

#3  
#3a  
cost <- matrix(c(0,1,4,0),nrow = 2, ncol = 2)  
cost

## [,1] [,2]  
## [1,] 0 4  
## [2,] 1 0

miss\_cost <- performance(pred1, "cost", cost.fp = 4, cost.fn = 1)  
cutoff\_new <- pred1@cutoffs[[1]][which.min(miss\_cost@y.values[[1]])]  
cutoff\_new

## 13   
## 0.8219539

#3b.  
flag\_train\_table <- ifelse(predicted\_train\_logistic > 0.8219539,1,0)  
flag\_train\_table

## 405 52 467 756 443 187 291 970 814 702 120 557 992 7 596   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 505 106 277 672 323 276 797 142 647 875 394 642 358 782 896   
## 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0   
## 905 59 750 444 8 811 202 16 827 558 554 987 200 497 176   
## 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0   
## 1 796 35 930 914 749 971 595 387 679 165 309 577 643 83   
## 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0   
## 177 542 897 864 904 437 64 862 209 583 356 257 340 178 665   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 417 393 888 194 508 501 350 407 582 459 993 704 860 735 889   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 189 234 271 742 442 164 490 19 512 92 677 482 472 565 977   
## 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0   
## 132 960 266 563 496 755 129 131 261 420 326 502 831 303 241   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 694 117 708 853 288 687 867 917 983 75 919 155 249 72 605   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 41 656 538 13 824 632 239 355 644 395 297 754 33 758 58   
## 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0   
## 697 614 705 191 974 77 367 10 597 292 947 87 217 243 168   
## 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0   
## 481 96 378 69 716 946 805 30 662 714 495 741 95 295 201   
## 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0   
## 503 553 89 957 103 710 579 858 198 494 886 912 318 149 158   
## 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0   
## 521 838 49 620 452 649 608 932 116 302 890 274 769 99 763   
## 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0   
## 806 18 798 870 907 743 840 385 375 130 118 341 379 956 604   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 135 121 62 670 279 661 631 663 125 425 465 57 5 711 569   
## 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0   
## 377 854 232 411 794 79 757 97 31 60 145 724 65 101 479   
## 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0   
## 206 675 240 114 441 617 636 954 594 180 869 306 174 771 461   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 413 267 576 78 253 37 793 871 91 872 921 803 492 474 354   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 328 812 150 143 865 874 828 859 12 686 264 868 449 219 937   
## 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0   
## 799 207 428 46 795 304 684 822 813 585 564 816 124 182 400   
## 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0   
## 601 616 507 781 252 739 1000 34 373 898 352 856 22 821 113   
## 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0   
## 440 21 696 535 93 195 786 847 282 475 29 598 365 473 727   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0   
## 692 27 629 560 924 14 846 701 487 327 693 273 768 285 972   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 933 790 422 20 88 86 419 707 994 259 296 624 286 360 849   
## 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0   
## 668 320 366 345 543 934 751 717 509 115 804 84 269 235 567   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 311 408 559 528 170 736 380 215 183 368 196 48 154 404 674   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 928 4 278 878 730 533 830 943 695 248 338 144 532 547 410   
## 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0   
## 468 386 544 551 439 262 944 2 107 634 774 593 655 169 251   
## 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0   
## 832 204 927 901 715 635 342 280 185 159 357 920 562 747 851   
## 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0   
## 762 247 906 537 918 740 765 651 900 746 775 436 767 685 388   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 381 516 788 100 398 861 607 402 362 721 268 119 916 409 819   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 991 166 464 584 712 809 450 446 637 141 531 984 800 36 996   
## 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0   
## 753 950 456 689 56 915 778 941 477 523 225 275 137 884 639   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 344 725 361 855 24 842 817 949 147 963 223 783 586 432 431   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 401 321 718 438 336 330 764 299 329 26 81 214 654 246 80   
## 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0   
## 913 384 986 499 731 324 534 457 517 455 923 843 820 709 307   
## 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0   
## 975 71 55 733 719 646 39 529 506 581 626 903 980 334 808   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 94 623 539 343 945 426 175 462 648 776 290 810 926 289 484   
## 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0   
## 371 885 70 844 109 737 761 224 606 826 738 190 163 211 152   
## 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0   
## 85 825 396 552 493 641 66 588 298 392 619 541 205 985 127   
## 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0   
## 162 936 669 337 333 50 453 602 348 283 848 530 700 683 74   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 627 953 300 199 748 979 390 591 802 766 186 908 525 752 447   
## 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0   
## 638 688 305 549 976 659 485 399 213 729 17 780 44 339 640   
## 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0   
## 734 807 108 837 578 966 678 791 787 599 469 720 658 834 256   
## 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0   
## 964 134 25 471 965 836 931 316 231 179 105 955 676 161 698   
## 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0   
## 770 233 51 229 978 666 466 435 284 228   
## 0 0 0 0 0 0 0 0 0 1

confusion\_logistic\_train <- table(as.numeric(train$L\_PREF),flag\_train\_table)  
confusion\_logistic\_train

## flag\_train\_table  
## 0 1  
## 0 565 2  
## 1 86 47

flag\_test\_table <- ifelse(predicted\_test\_logistic >0.8219539,1,0)  
flag\_test\_table

## 3 6 9 11 15 23 28 32 38 40 42 43 45 47 53 54 61 63   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 67 68 73 76 82 90 98 102 104 110 111 112 122 123 126 128 133 136   
## 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0   
## 138 139 140 146 148 151 153 156 157 160 167 171 172 173 181 184 188 192   
## 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0   
## 193 197 203 208 210 212 216 218 220 221 222 226 227 230 236 237 238 242   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 244 245 250 254 255 258 260 263 265 270 272 281 287 293 294 301 308 310   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0   
## 312 313 314 315 317 319 322 325 331 332 335 346 347 349 351 353 359 363   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 364 369 370 372 374 376 382 383 389 391 397 403 406 412 414 415 416 418   
## 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0   
## 421 423 424 427 429 430 433 434 445 448 451 454 458 460 463 470 476 478   
## 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0   
## 480 483 486 488 489 491 498 500 504 510 511 513 514 515 518 519 520 522   
## 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0   
## 524 526 527 536 540 545 546 548 550 555 556 561 566 568 570 571 572 573   
## 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 574 575 580 587 589 590 592 600 603 609 610 611 612 613 615 618 621 622   
## 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0   
## 625 628 630 633 645 650 652 653 657 660 664 667 671 673 680 681 682 690   
## 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 691 699 703 706 713 722 723 726 728 732 744 745 759 760 772 773 777 779   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 784 785 789 792 801 815 818 823 829 833 835 839 841 845 850 852 857 863   
## 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 866 873 876 877 879 880 881 882 883 887 891 892 893 894 895 899 902 909   
## 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0   
## 910 911 922 925 929 935 938 939 940 942 948 951 952 958 959 961 962 967   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 968 969 973 981 982 988 989 990 995 997 998 999   
## 0 0 0 0 1 0 0 0 0 0 0 1

confusion\_logistic\_test <- table(as.numeric(test$L\_PREF),flag\_test\_table)  
confusion\_logistic\_test

## flag\_test\_table  
## 0 1  
## 0 238 4  
## 1 37 21

misclassif\_cost\_training <- confusion\_logistic\_train \* cost  
misclassif\_cost\_training

## flag\_train\_table  
## 0 1  
## 0 0 8  
## 1 86 0

sum(misclassif\_cost\_training)

## [1] 94

misclassif\_cost\_testing <- confusion\_logistic\_test \* cost  
misclassif\_cost\_testing

## flag\_test\_table  
## 0 1  
## 0 0 16  
## 1 37 0

sum(misclassif\_cost\_testing)

## [1] 53

#3c  
flag\_train\_table\_new <- ifelse(predicted\_train\_logistic > 0.4625541,1,0)  
  
confusion\_logistic\_train\_old <- table(as.numeric(train$L\_PREF),flag\_train\_table\_new)  
confusion\_logistic\_train\_old

## flag\_train\_table\_new  
## 0 1  
## 0 543 24  
## 1 40 93

flag\_test\_table\_new<- ifelse(predicted\_test\_logistic >0.4625541,1,0)  
confusion\_logistic\_test\_old <- table(as.numeric(test$PREF),flag\_test\_table\_new)  
confusion\_logistic\_test\_old

## flag\_test\_table\_new  
## 0 1  
## 1 21 37  
## 2 225 17

misclassification\_cost\_training\_old <- confusion\_logistic\_train\_old \* cost  
misclassification\_cost\_training\_old

## flag\_train\_table\_new  
## 0 1  
## 0 0 96  
## 1 40 0

sum(misclassification\_cost\_training\_old)

## [1] 136

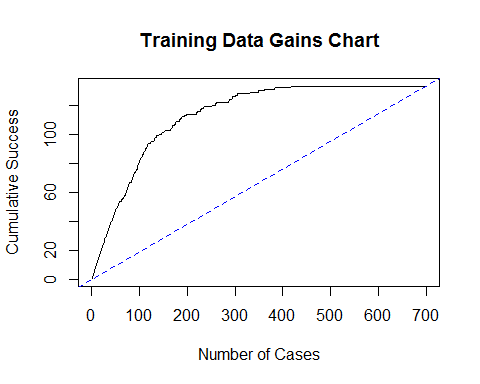
misclassification\_cost\_testing\_old <- confusion\_logistic\_test\_old \* cost  
misclassification\_cost\_testing\_old

## flag\_test\_table\_new  
## 0 1  
## 1 0 148  
## 2 225 0

sum(misclassification\_cost\_testing\_old)

## [1] 373

#4.  
#Training Data  
actual <- train$L\_PREF  
df\_train <- data.frame(predicted\_train\_logistic,actual)  
df\_train\_sort <- df\_train[order(-predicted\_train\_logistic),]  
df\_train\_sort$Gains <- cumsum(df\_train\_sort$actual)  
plot(df\_train\_sort$Gains,type="n",main="Training Data Gains Chart",xlab="Number of Cases",ylab="Cumulative Success")  
lines(df\_train\_sort$Gains)  
abline(0,sum(df\_train\_sort$actual)/nrow(df\_train\_sort),lty = 2, col="blue")



###################################################################################  
  
#Test Data  
actual <- test$L\_PREF  
df\_test <- data.frame(predicted\_test\_logistic,actual)  
df\_test\_sort <- df\_test[order(-predicted\_test\_logistic),]  
df\_test\_sort$Gains <- cumsum(df\_test\_sort$actual)  
plot(df\_test\_sort$Gains,type="n",main="Test Data Gains Chart",xlab="Number of Cases",ylab="Cumulative Success")  
lines(df\_test\_sort$Gains)  
abline(0,sum(df\_test\_sort$actual)/nrow(df\_test\_sort),lty = 2, col="green")

