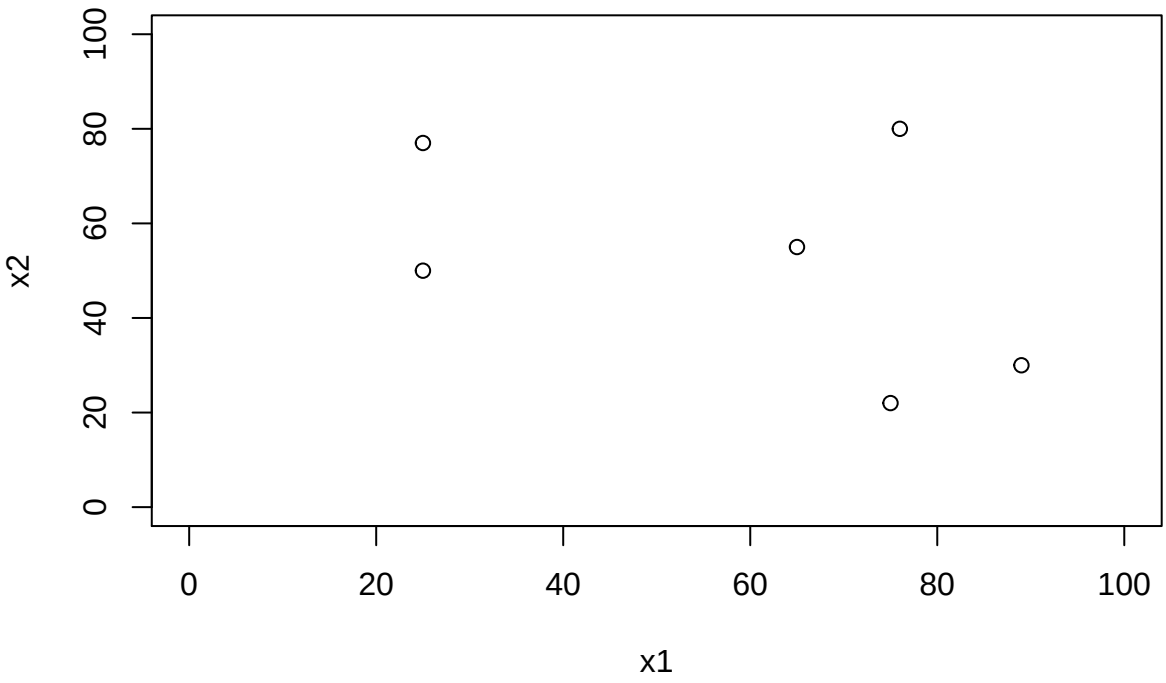
Tree-hw-Ziyi Bai

Ziyi Bai

3/3/2021

8.1

```
data <- data.frame(c(25,25,75,76,65,89),c(77,50,22,80,55,30))
plot(data, xlim=c(0,100),ylim=c(0,100),xlab="x1",ylab="x2")
```



8.2

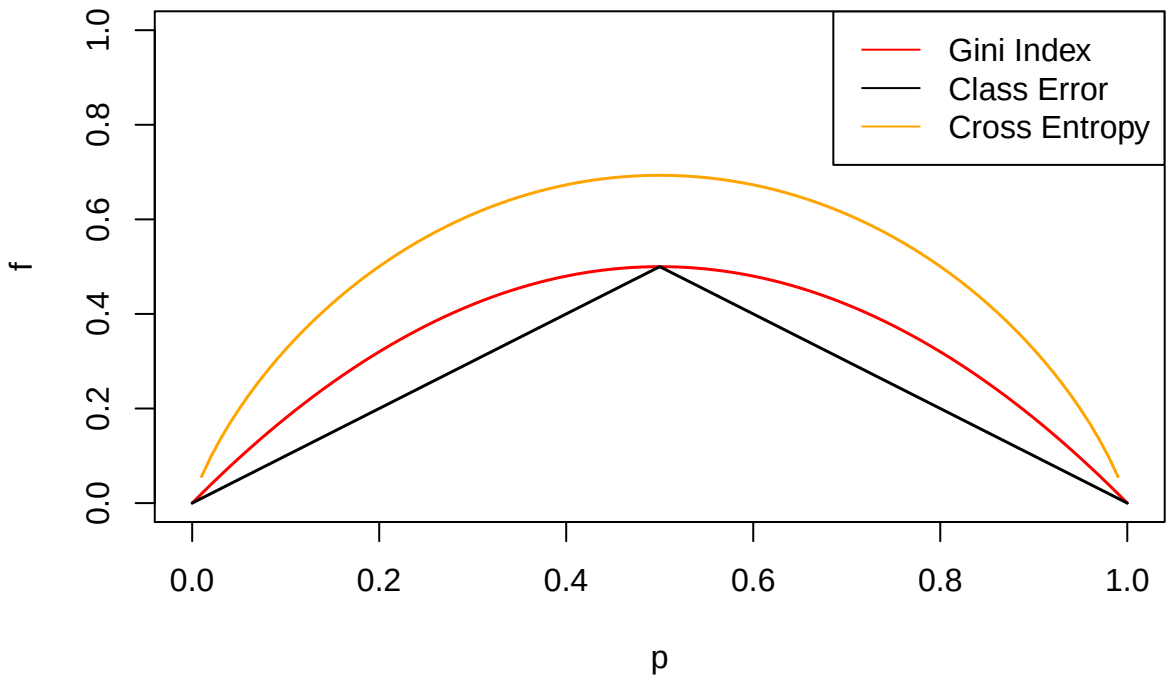
8.3

```
p <- seq(0,1,0.01)

gini <- 2*p*(1-p)
error <- 1-pmax(p,1-p)
entropy <- -(p*log(p)+(1-p)*log(1-p))

plot(NA,xlim = c(0,1),ylim = c(0,1),xlab = "p",ylab = "f")
lines(p,gini,type = "l",col="red",lwd=1.5)
lines(p,error,type = "l",col="black",lwd=1.5)
lines(p,entropy,type="l",col="orange",lwd=1.5)
```

```
legend(x="topright",legend = c("Gini Index","Class Error","Cross Entropy"),col=c("red","black","orange")
```



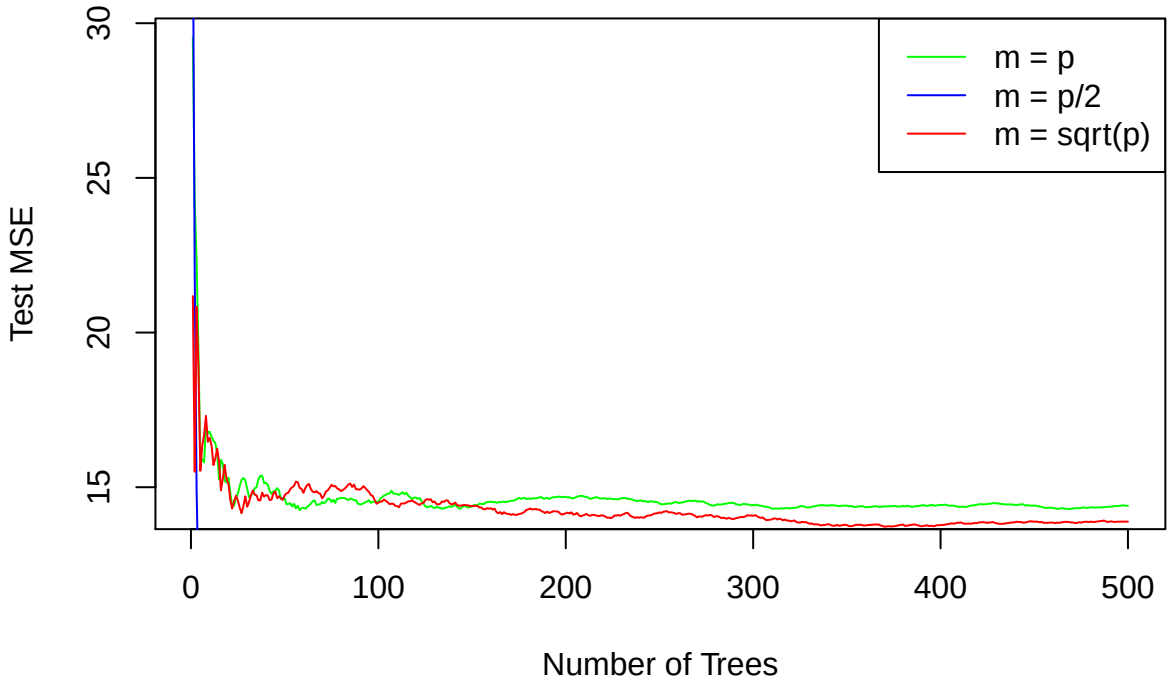
8.5

Based on majority vote approach, the result is red; based on average probability, the result is green.

8.7

```
data("Boston")
set.seed(9)
train <- sample(1:nrow(Boston),nrow(Boston)/2)
Boston_train <- Boston[train,-14]
Boston_test <- Boston[-train,-14]
y_train <- Boston[train,14]
y_test <- Boston[-train,14]

rf1 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = ncol(Boston)
rf2 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = (ncol(Boston
rf3 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = sqrt(ncol(Bos
plot(1:500, rf1$test$mse, type = "l", col = "green", xlab = "Number of Trees", ylab = "Test MSE")
lines(1:500, rf2$test$mse, type = "l", col = "blue")
lines(1:500, rf3$test$mse, type = "l", col = "red")
legend(x = "topright", c("m = p", "m = p/2", "m = sqrt(p)"), col = c("green", "blue", "red"), lty = 1)
```



8.8

##(a)

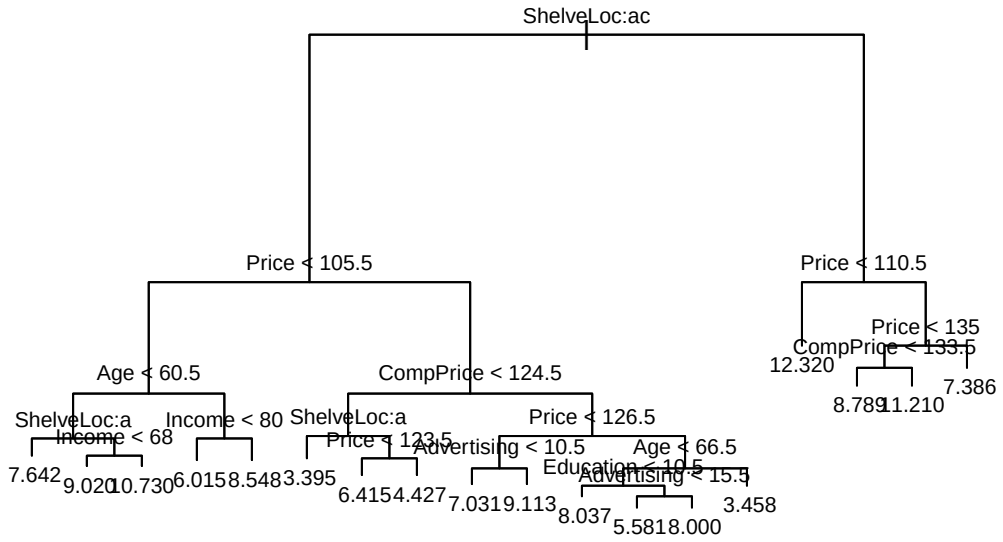
```
data("Carseats")
set.seed(9)
subs <-sample(1:nrow(Carseats),nrow(Carseats)*0.7)
car_train <- Carseats[subs, ]
car_test <- Carseats[-subs, ]
```

##(b)

```
rtree <- tree(Sales~.,data=car_train)
summary(rtree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = car_train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising" "Education"
## Number of terminal nodes:  18
## Residual mean deviance:  2.621 = 686.6 / 262
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.2110 -1.0330 -0.0156  0.0000  0.9664  4.2340
```

```
plot(rtree)
text(rtree,cex=0.65)
```

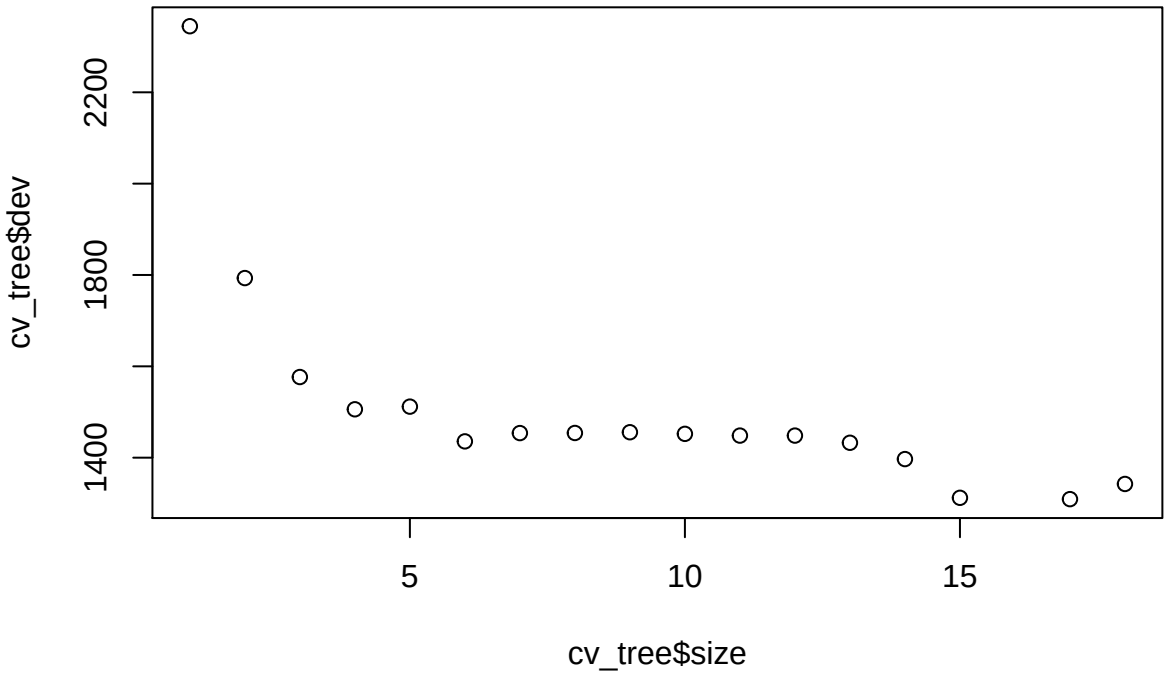



```
#MSE
pred_rtree <- predict(rtree,car_test)
mse_rtree <- mean((car_test$Sales-pred_rtree)^2)
print(paste0("The test MSE for the regression tree is:", mse_rtree))
```

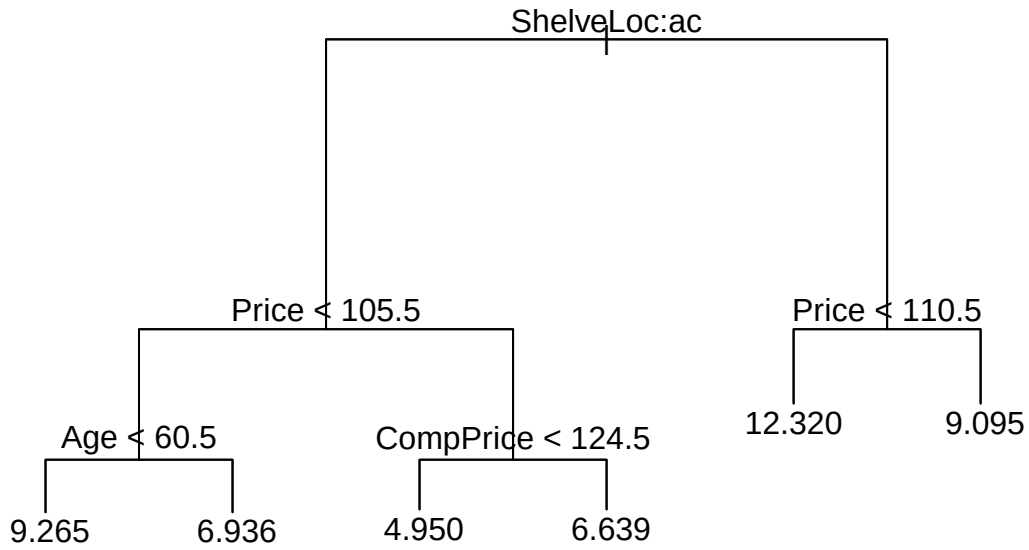
```
## [1] "The test MSE for the regression tree is:4.6376957387513"
```

##(c)

```
cv_tree <- cv.tree(rtree)
plot(cv_tree$size,cv_tree$dev)
```




```
prune_rtree <- prune.tree(rtree,best=6)
plot(prune_rtree)
text(prune_rtree)
```




```
prune_pred <- predict(prune_rtree,car_test)
prune_mse <- mean((prune_pred-car_test$Sales)^2)
print(paste0("The test MSE for the pruned tree is:"),prune_mse)
```

```
## [1] "The test MSE for the pruned tree is:"
```

##(d)

```
car_bag <- randomForest(Sales~.,data=car_train,mtry=10,importance=T,ntree=500)
pred_bag <- predict(car_bag,car_test)
bag_mse <- mean((pred_bag-car_test$Sales)^2)

print(paste0("The test MSE for bagging method is:", bag_mse))
```

```
## [1] "The test MSE for bagging method is:2.06113398392704"
```

```
importance(car_bag)
```

```
##                %IncMSE IncNodePurity
## CompPrice   32.2046620     277.99211
## Income       8.8431792     112.30771
## Advertising 20.7348575     179.92226
## Population  -1.3035020      80.89917
## Price       71.0352250     695.62717
## ShelveLoc   68.3097260     651.42916
## Age         17.3234447     194.22526
## Education    1.4399098      56.31854
## Urban       -0.5457821      10.93868
## US           5.4387117      17.93217
```

##(e)

```
rf_mse <- c()
for (i in 1:10) {
  car_rf <- randomForest(Sales ~ ., data = car_train, mtry = i, importance = TRUE, ntree = 500)
  pred_rf <- predict(car_rf, car_test)
  rf_mse[i] <- mean((pred_rf - car_test$Sales)^2)
}
#Best model
```

```r
which.min(rf_mse)
```

```
## [1] 9
```

```r
#Minimum MSE
rf_mse[which.min(rf_mse)]
```

```
## [1] 2.067719
```

```r
importance(car_rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   33.1964870    273.636132
## Income       6.2208676    119.855006
## Advertising 21.1559605    184.384037
## Population  -1.3882319     80.033794
## Price       67.5697148    696.008657
## ShelveLoc   67.5922455    642.087660
## Age         16.8487080    190.268240
## Education    1.0094581     52.403296
## Urban        0.7675524      9.238638
## US           6.3153951     17.320637
```

8.11

##(a)

```r
data("Caravan")
Caravan$Purchase <- ifelse(Caravan$Purchase=="No",0,1)
crv_train <- Caravan[1:1000,]
crv_test <- Caravan[1001:5822,]
```

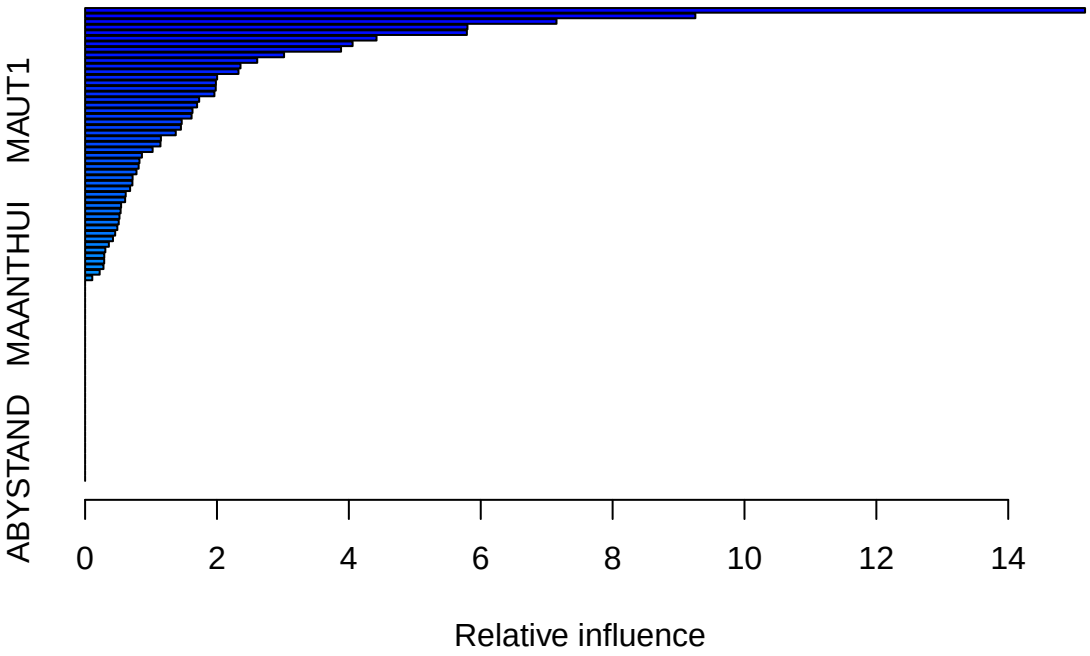
##(b)

```r
set.seed(9)
boost <- gbm(Purchase~.,data=crv_train,shrinkage = 0.01,n.trees = 1000, distribution = "bernoulli")
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 50: PVRAAUT has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 71: AVRAAUT has no variation.
```

```r
kable(summary(boost),row.names=F)
```



var	rel.inf
PPERSAUT	15.1650119
MKOOPKLA	9.2549208
MOPLHOOG	7.1491523
MBERMIDD	5.7988033
PBRAND	5.7897473
MGODGE	4.4197200
MINK3045	4.0569774
ABRAND	3.8816596
MOSTYPE	3.0173120
MSKA	2.6104471
MSKC	2.3567316
MAUT2	2.3263968
PWAPART	2.0023871
MINKGEM	1.9838691
MBERARBG	1.9814157
MGODPR	1.9612263
MGODOV	1.7300166
MFWEKIND	1.6986371
MAUT1	1.6287004
PBYSTAND	1.6148436
MSKB1	1.4654283
MRELGE	1.4532182
MBERHOOG	1.3751342
MHHUUR	1.1499571
MRELOV	1.1429171
APERSAUT	1.0241970
MOSHOOFD	0.8617721
MINK7512	0.8244418
MFGEKIND	0.8122327
MSKD	0.7794502
MGODRK	0.7204269

var	rel.inf
MAUT0	0.7176358
MINKM30	0.6825040
MHKOOP	0.6160159
MOPLMIDD	0.6069681
MBERARBO	0.5460070
MINK123M	0.5388234
MBERBOER	0.5211953
MGEMOMV	0.5101719
MGEMLEEF	0.4889504
MINK4575	0.4563592
MFALLEEN	0.4226122
PMOTSCO	0.3601726
MSKB2	0.3067847
MZFONDS	0.2906486
MZPART	0.2897102
MOPLLAAG	0.2787625
PLEVEN	0.2207685
MRELSA	0.1087580
MAANTHUI	0.0000000
MBERZELF	0.0000000
PWABEDR	0.0000000
PWALAND	0.0000000
PBESAUT	0.0000000
PVRAAUT	0.0000000
PAANHANG	0.0000000
PTRACTOR	0.0000000
PWERKT	0.0000000
PBROM	0.0000000
PPERSONG	0.0000000
PGEZONG	0.0000000
PWAOREG	0.0000000
PZEILPL	0.0000000
PPLEZIER	0.0000000
PFIETS	0.0000000
PINBOED	0.0000000
AWAPART	0.0000000
AWABEDR	0.0000000
AWALAND	0.0000000
ABESAUT	0.0000000
AMOTSCO	0.0000000
AVRAAUT	0.0000000
AAANHANG	0.0000000
ATRACTOR	0.0000000
AWERKT	0.0000000
ABROM	0.0000000
ALEVEN	0.0000000
APERSONG	0.0000000
AGEZONG	0.0000000
AWAOREG	0.0000000
AZEILPL	0.0000000
APLEZIER	0.0000000
AFIETS	0.0000000

var	rel.inf
AINBOED	0.0000000
ABYSTAND	0.0000000

##(c)

```
pred_boost <- predict(boost,crv_test,n.trees = 1000,type = "response")
boost_pred <- ifelse(pred_boost>0.2,1,0)
table(crv_test$Purchase,boost_pred)
```

```
##    boost_pred
##        0    1
##   0 4415  118
##   1  253   36
```

```
crv_glm <- glm(Purchase~.,data=crv_train,family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred_glm <- predict(crv_glm,crv_test,type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
glm_pred <- ifelse(pred_glm>0.2, 1,0)
table(crv_test$Purchase,glm_pred)
```

```
##    glm_pred
##        0    1
##   0 4183  350
##   1  231   58
```

36/(36+118)=0.2337 58/(58+350)=0.1421