Blood Donations

Bridget Bertoni

April 12, 2018

```r
# read in data
train_data=read.csv(file="/Users/bbertoni/Desktop/github/Blood_Donations/training_data.csv",header=T)
test_data=read.csv(file="/Users/bbertoni/Desktop/github/Blood_Donations/test_data.csv",header=T)

names(train_data)=c("ID","Last_donation","Num_donations","Volume","First_donation","Made_donation")
names(test_data)=c("ID","Last_donation","Num_donations","Volume","First_donation")

head(train_data)
```

```
##    ID Last_donation Num_donations Volume First_donation Made_donation
## 1 619             2            50  12500             98             1
## 2 664             0            13   3250             28             1
## 3 441             1            16   4000             35             1
## 4 160             2            20   5000             45             1
## 5 358             1            24   6000             77             0
## 6 335             4             4   1000              4             0
```

```r
head(test_data)
```

```
##    ID Last_donation Num_donations Volume First_donation
## 1 659             2            12   3000             52
## 2 276            21             7   1750             38
## 3 263             4             1    250              4
## 4 303            11            11   2750             38
## 5  83             4            12   3000             34
## 6 500             3            21   5250             42
```

```r
#train_data$Made_donation=as.factor(train_data$Made_donation)
#test_data$Made_donation=as.factor(test_data$Made_donation)

# split training data into a training set and a validation set
set.seed(333)
train=sample(1:nrow(train_data),0.7*nrow(train_data),replace=F)
val_data = train_data[-train,]
train_data = train_data[train,]

# check for missing or strange values
sum(is.na(train_data))
```

```
## [1] 0
```

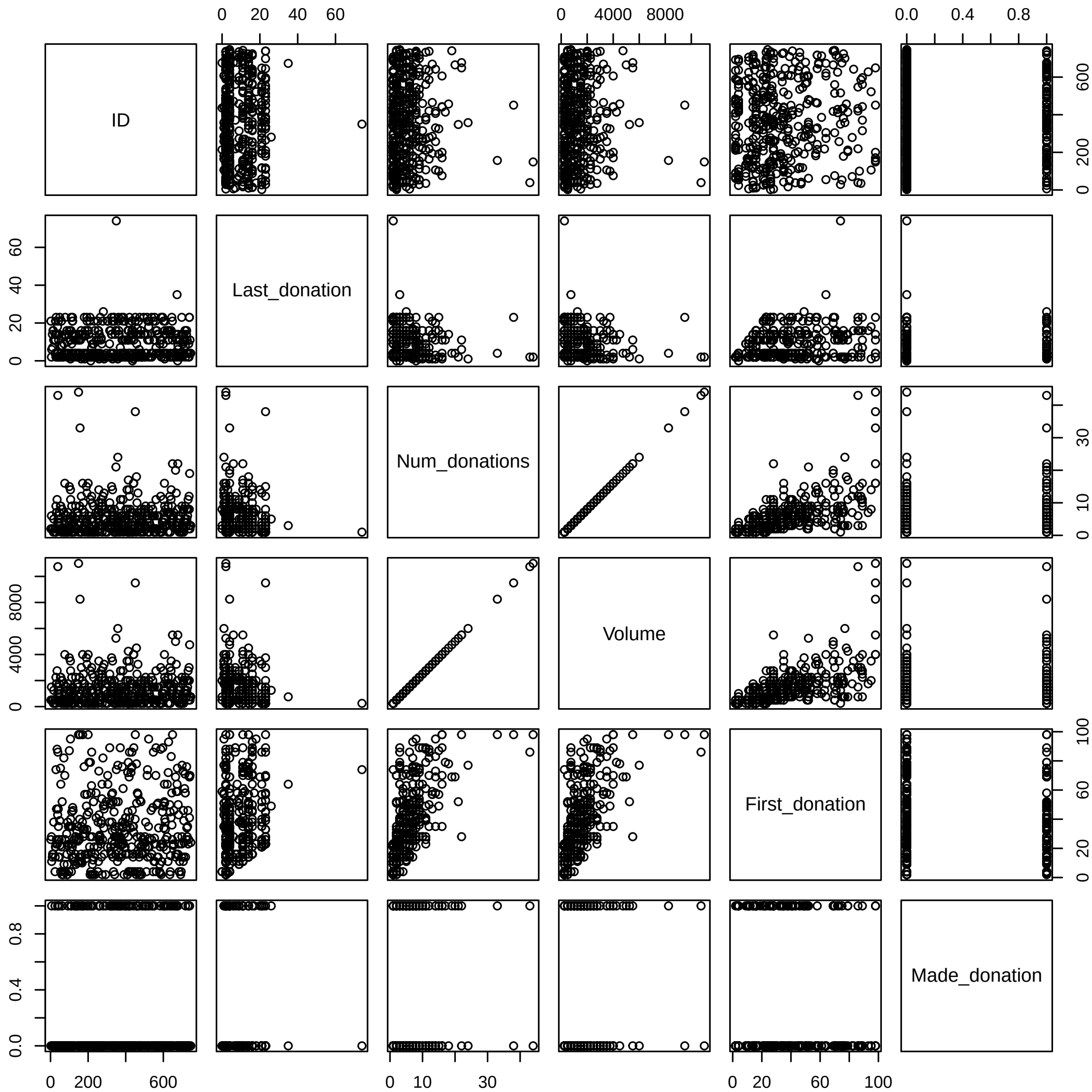
```r
sum(is.na(val_data))# no missing values
```

```
## [1] 0
```

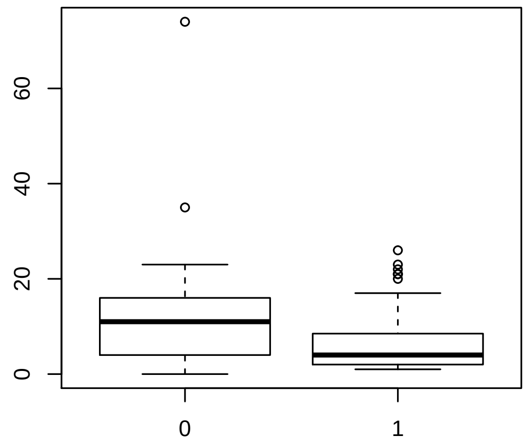
```r
sum(is.na(test_data)) # missing values
```

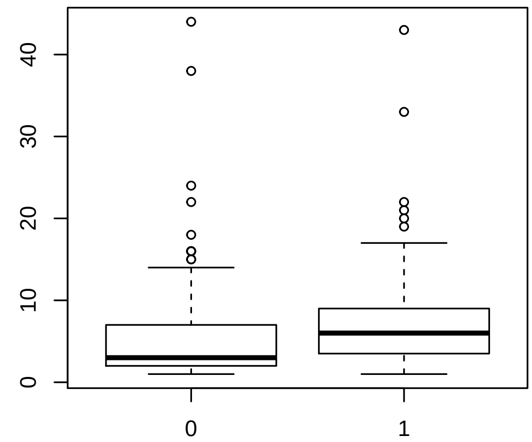
```
## [1] 0
```

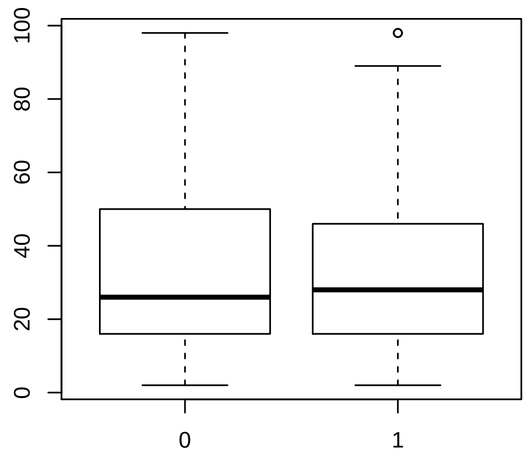
```r
plot(train_data)
```

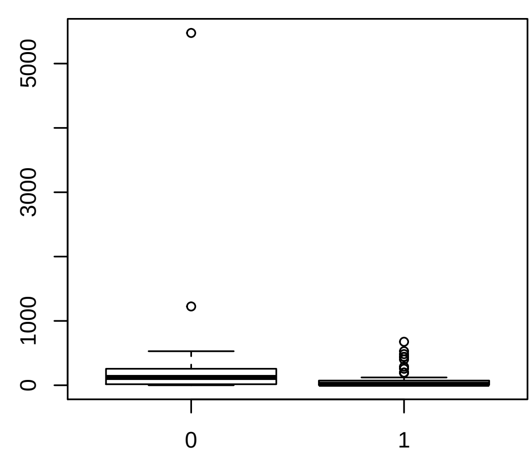



```r
par(mfrow=c(2,2))
plot(as.factor(train_data$Made_donation),train_data$Last_donation)
plot(as.factor(train_data$Made_donation),train_data$Num_donations)
plot(as.factor(train_data$Made_donation),train_data$First_donation)
plot(as.factor(train_data$Made_donation),train_data$Last_donation*train_data$Last_donation)
```

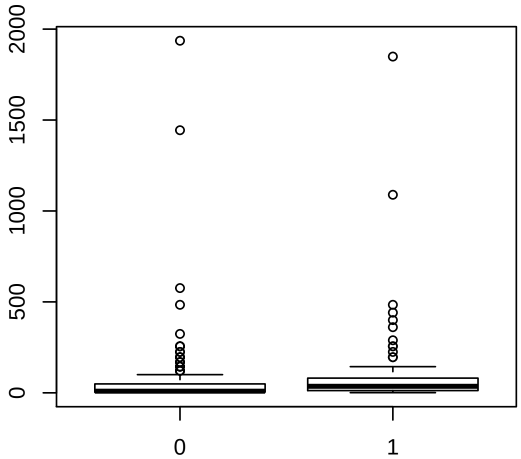


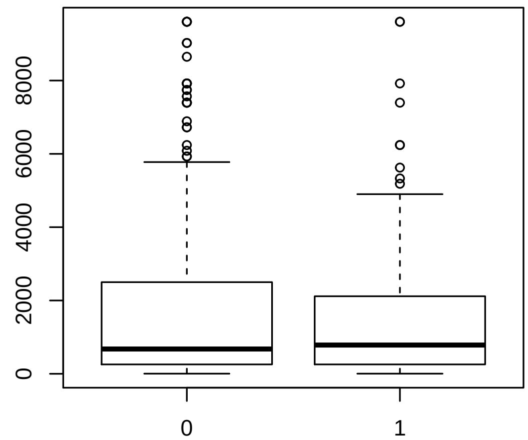







```
plot(as.factor(train_data$Made_donation),train_data$Num_donation*train_data$Num_donation)
plot(as.factor(train_data$Made_donation),train_data$First_donation*train_data$First_donation)
par(mfrow=c(1,1))
```






```r
cor(train_data) # correlation between volume and the number of donations is 1
```

```
##                         ID Last_donation Num_donations      Volume
## ID              1.00000000    0.02180929    0.02167073  0.02167073
## Last_donation   0.02180929    1.00000000   -0.16150193 -0.16150193
## Num_donations   0.02167073   -0.16150193    1.00000000  1.00000000
## Volume          0.02167073   -0.16150193    1.00000000  1.00000000
## First_donation  0.10391260    0.17791940    0.64740429  0.64740429
## Made_donation   0.04101195   -0.24721682    0.20707525  0.20707525
##                First_donation Made_donation
## ID                0.103912600   0.041011949
## Last_donation     0.177919401  -0.247216821
## Num_donations     0.647404293   0.207075245
## Volume            0.647404293   0.207075245
```

```
## First_donation    1.000000000   0.008180282
## Made_donation     0.008180282   1.000000000
```

```r
# fit basic logistic regression
glm.fit=glm(Made_donation~Last_donation+Num_donations+First_donation,data=train_data,
            family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Made_donation ~ Last_donation + Num_donations +
##     First_donation, family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3459  -0.7959  -0.5454  -0.3334   2.4440
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.715501   0.240869  -2.970  0.00297 **
## Last_donation  -0.082632   0.021242  -3.890  0.00010 ***
## Num_donations   0.109057   0.033638   3.242  0.00119 **
## First_donation -0.012572   0.007533  -1.669  0.09513 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 440.16  on 402  degrees of freedom
## Residual deviance: 396.46  on 399  degrees of freedom
## AIC: 404.46
##
## Number of Fisher Scoring iterations: 5
```

```r
#plot(glm.fit)
glm.probs=predict(glm.fit,val_data,type="response")
glm.pred=rep(0,nrow(val_data))
glm.pred[glm.probs>0.5]=1
table(glm.pred,val_data$Made_donation) # confusion matrix
```

```
##
## glm.pred   0   1
##        0 129  38
##        1   1   5
```

```r
mean(glm.pred==val_data$Made_donation)
```

```
## [1] 0.7745665
```

```r
# log loss
-(1/nrow(val_data))*( sum(val_data$Made_donation*log(glm.probs)) +
    sum((1-val_data$Made_donation)*log(1-glm.probs)) )
```

```
## [1] 0.4671785
```

```r
# calculate predictions using all of the training data
final_train_data=rbind(train_data,val_data)
```

```r
glm.fit=glm(Made_donation~Last_donation+Num_donations+First_donation,data=train_data,
            family=binomial)
glm.probs=predict(glm.fit,test_data,type="response")
out=data.frame(test_data$ID,glm.probs)
names(out)=c("","Made Donation in March 2007")
write.csv(out,file="logreg_2018_04_16.csv",row.names=FALSE)

# fit basic logistic regression, drop first donation
glm.fit=glm(Made_donation~Last_donation+Num_donations,data=train_data,
            family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Made_donation ~ Last_donation + Num_donations,
##     family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0754  -0.8127  -0.5449  -0.3272   2.4324
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.82025    0.23322  -3.517 0.000436 ***
## Last_donation -0.09346    0.02029  -4.606  4.1e-06 ***
## Num_donations  0.06903    0.02210   3.124 0.001785 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 440.16  on 402  degrees of freedom
## Residual deviance: 399.42  on 400  degrees of freedom
## AIC: 405.42
##
## Number of Fisher Scoring iterations: 5
```

```r
#plot(glm.fit)
glm.probs=predict(glm.fit,val_data,type="response")
glm.pred=rep(0,nrow(val_data))
glm.pred[glm.probs>0.5]=1
table(glm.pred,val_data$Made_donation) # confusion matrix
```

```
##
## glm.pred   0   1
##        0 128  39
##        1   2   4
```

```r
mean(glm.pred==val_data$Made_donation)
```

```
## [1] 0.7630058
```

```r
# log loss
-(1/nrow(val_data))*( sum(val_data$Made_donation*log(glm.probs)) +
    sum((1-val_data$Made_donation)*log(1-glm.probs)) )
```

```
## [1] 0.4820182
```

```
# calculate predictions using all of the training data
final_train_data=rbind(train_data,val_data)
glm.fit=glm(Made_donation~Last_donation+Num_donations,data=train_data,
            family=binomial)
glm.probs=predict(glm.fit,test_data,type="response")
out=data.frame(test_data$ID,glm.probs)
names(out)=c("","Made Donation in March 2007")
write.csv(out,file="logreg_nofirstdon_2018_04_16.csv",row.names=FALSE)

# fit logistic regression with interaction terms
glm.fit=glm(Made_donation~Last_donation+Num_donations+First_donation+Last_donation:Num_donations,data=t
            family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Made_donation ~ Last_donation + Num_donations +
##     First_donation + Last_donation:Num_donations, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6845  -0.7724  -0.5472  -0.3417   2.4294
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -0.881040   0.290702  -3.031  0.00244 **
## Last_donation               -0.062340   0.028078  -2.220  0.02640 *
## Num_donations                0.140875   0.046997   2.998  0.00272 **
## First_donation              -0.013458   0.007744  -1.738  0.08221 .
## Last_donation:Num_donations -0.003388   0.003301  -1.026  0.30473
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 440.16  on 402  degrees of freedom
## Residual deviance: 395.29  on 398  degrees of freedom
## AIC: 405.29
##
## Number of Fisher Scoring iterations: 5
```

```
#plot(glm.fit)
glm.probs=predict(glm.fit,val_data,type="response")
glm.pred=rep(0,nrow(val_data))
glm.pred[glm.probs>0.5]=1
table(glm.pred,val_data$Made_donation) # confusion matrix
```

```
##
## glm.pred   0   1
##        0 126  36
##        1   4   7
```

```r
mean(glm.pred==val_data$Made_donation)
```

```
## [1] 0.7687861
```

```r
# log loss
-(1/nrow(val_data))*( sum(val_data$Made_donation*log(glm.probs)) +
    sum((1-val_data$Made_donation)*log(1-glm.probs)) )
```

```
## [1] 0.4631748
```

```r
# calculate predictions using all of the training data
final_train_data=rbind(train_data,val_data)
glm.fit=glm(Made_donation~Last_donation+Num_donations+First_donation+Last_donation:Num_donations,data=t
            family=binomial)
glm.probs=predict(glm.fit,test_data,type="response")
out=data.frame(test_data$ID,glm.probs)
names(out)=c("","Made Donation in March 2007")
write.csv(out,file="logreg_int_2018_04_16.csv",row.names=FALSE)
```

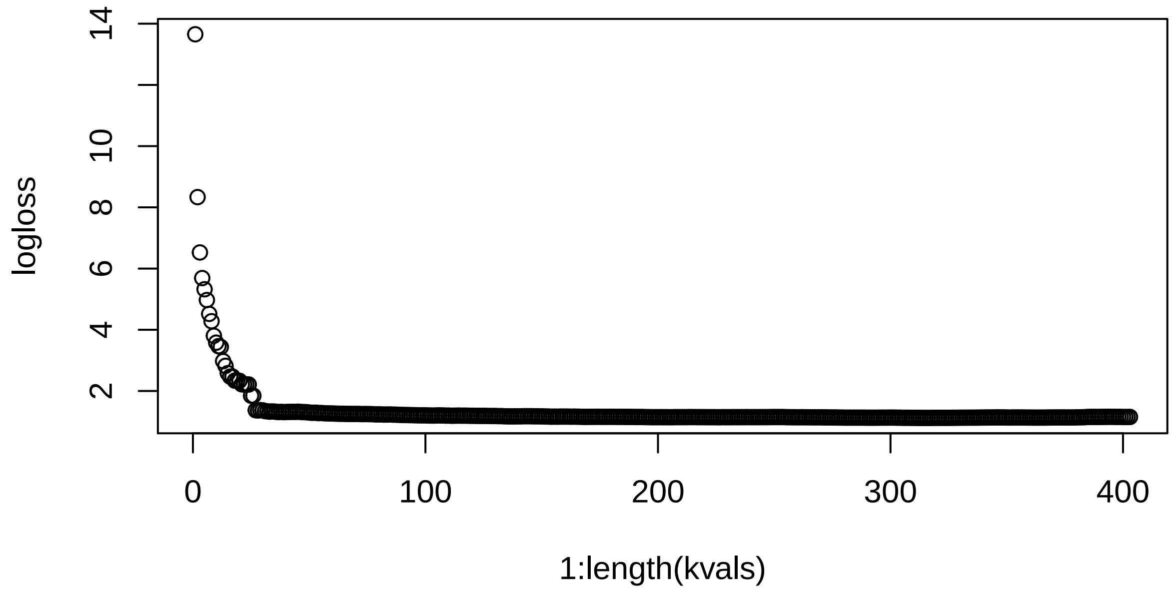
```r
# fit KNN, choose k to minimize the cost on the validation set
library(class)
train.X=as.matrix(train_data[,-c(1,4,6)],nrow=nrow(train_data),ncol=ncol(train_data)-3)
val.X=as.matrix(val_data[,-c(1,4,6)],nrow=nrow(val_data),ncol=ncol(val_data)-3)
set.seed(111)
kvals=seq(1,nrow(train_data))
logloss=rep(NA,length(kvals))
for (k in 1:length(kvals)){
  knn.pred=knn(train.X,val.X,train_data$Made_donation,k=k,prob=TRUE)
  knn.prob=abs(attr(knn.pred,"prob")-10^-10) # need to add a fudge factor to deal
                                             # with logs of 0 and 1
  logloss[k]=-(1/nrow(val_data))*( sum(val_data$Made_donation*log(knn.prob)) +
    sum((1-val_data$Made_donation)*log(1-knn.prob)) )
}
which.min(logloss)
```

```
## [1] 316
```

```r
logloss[which.min(logloss)]
```

```
## [1] 1.120105
```

```r
plot(1:length(kvals),logloss)
```




```
# choose k at the elbow, note small k is high variance, low bias:
plot(1:50,logloss[1:50])

# calculate predictions using all of the training data
k=27 # pick k = 27
final_train.X=rbind(train.X,val.X)
test.X=as.matrix(test_data[,-c(1,4)],nrow=nrow(test_data),ncol=ncol(test_data)-2)
knn.pred=knn(final_train.X,test.X,c(train_data$Made_donation,val_data$Made_donation),k=k,
             prob=TRUE)
knn.prob=abs(attr(knn.pred,"prob")-10^-10)
out=data.frame(test_data$ID,knn.prob)
names(out)=c("","Made Donation in March 2007")
write.csv(out,file="knn_2018_04_16.csv",row.names=FALSE)

# fit a random forest with bagging
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(200)
train_data$Made_donation=as.factor(train_data$Made_donation)
val_data$Made_donation=as.factor(val_data$Made_donation)
#test_data$Made_donation=as.factor(test_data$Made_donation)
m=ncol(train_data)-3
bag.data=randomForest(Made_donation~Last_donation+Num_donations+First_donation+Last_donation:Num_donati
bag.data
```

```
##
## Call:
##  randomForest(formula = Made_donation ~ Last_donation + Num_donations +      First_donation + Last_d
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 24.81%
## Confusion matrix:
##     0  1 class.error
## 0 271 37   0.1201299
## 1  63 32   0.6631579
```

```
importance(bag.data)
```

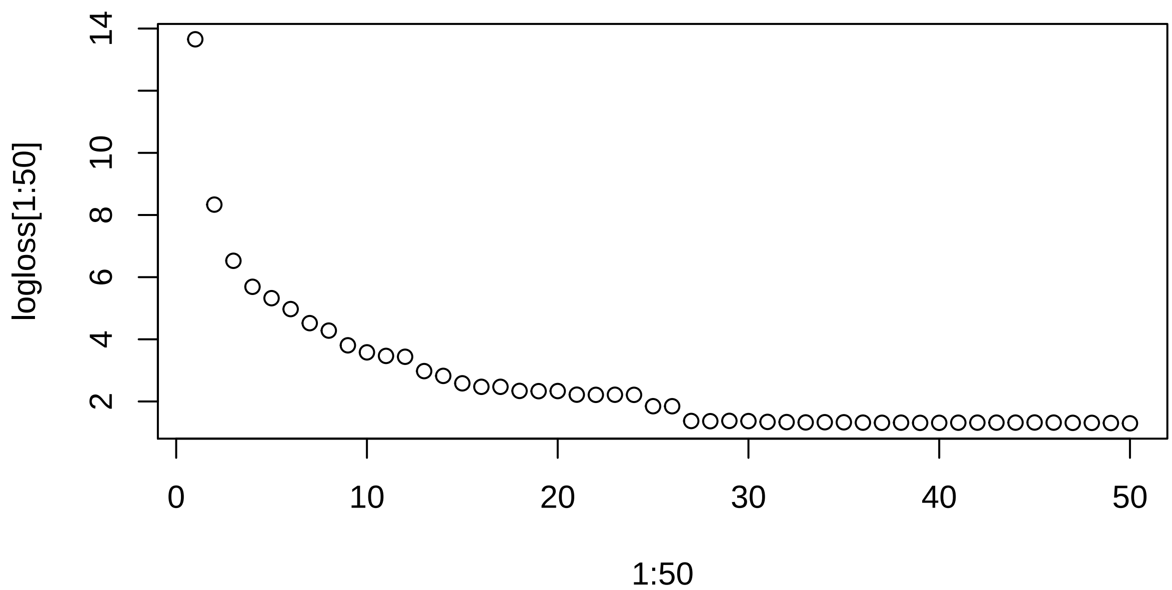
```
##                        0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Last_donation   7.907709 11.659565             13.06505         31.59352
## Num_donations  21.943140  9.541482             26.45239         41.02613
## First_donation 18.246722 -1.557938             17.11908         51.15227
```

```
probs=predict(bag.data,newdata=val_data,type="prob")[,2]+10^-10
# log loss
-(1/nrow(val_data))*( sum((as.numeric(val_data$Made_donation)-1)*log(probs)) +
    sum((1-(as.numeric(val_data$Made_donation)-1))*log(1-probs)) )
```

```
## [1] 1.13197
```

```
# fit a random forest with boosting
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```




```
lambdas=c(10^-5,10^-4,10^-3,10^-2,0.05,0.1,0.2,0.5)
logloss=rep(NA,length(lambdas))
train_data_exp=cbind(train_data,train_data$Last_donation*train_data$Num_donations)
names(train_data_exp)[7]="Last_num_int"
val_data_exp=cbind(val_data,val_data$Last_donation*val_data$Num_donations)
names(val_data_exp)[7]="Last_num_int"
for (i in 1:length(lambdas)){
  lambda=lambdas[i]
  boost.data=gbm(Made_donation~Last_donation+Num_donations+First_donation+Last_num_int,data=train_data_e
                distribution="bernoulli",n.trees=1000,shrinkage=lambda,interaction.depth=2)
  probs=predict(bag.data,newdata=val_data_exp,type="prob")[,2]+10^-10
  probs
  logloss[i]=-(1/nrow(val_data))*( sum((as.numeric(val_data$Made_donation)-1)*log(probs)) +
    sum((1-(as.numeric(val_data$Made_donation)-1))*log(1-probs)) )
}
logloss
```

```
## [1] 1.13197 1.13197 1.13197 1.13197 1.13197 1.13197 1.13197 1.13197
```