Kaggle Project

Bridget, Eva, and Annie

November 28, 2017

```
# read in the data
train_data=read.csv(file="/home/bridget/Dropbox/STATS202/kaggle/train.csv",header=T)
test_data=read.csv(file="/home/bridget/Dropbox/STATS202/kaggle/test.csv",header=T)

# remove ID
names(train_data)
```

```
##  [1] "ID"            "Rel.Compact"   "Surface.Area"  "Wall.Area"
##  [5] "Roof.Area"     "Height"        "Orientation"   "Glazing.Area"
##  [9] "Glazing.Distr" "Outcome"
```

```
train_data=train_data[,-1]

# change orientation and glazing distribution to factors
train_data$Orientation=as.factor(train_data$Orientation)
train_data$Glazing.Distr=as.factor(train_data$Glazing.Distr)

# boosted decision tree
library(gbm)
```

```
## Loading required package: survival
```

```
## Loading required package: lattice
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
set.seed(1)
train_sample=sample(1:nrow(train_data),0.7*nrow(train_data))
mse=matrix(rep(NA,5*10),nrow=10,ncol=5)
for (i in 1:10){
  k=0
  for (t in c(50,100,500,1000,5000)) {
    k=k+1
    boost=gbm(Outcome~.-Orientation,data=train_data[train_sample,],
                        distribution="gaussian",n.trees=t,interaction.depth=i,shrinkage=0.1)
    mse[i,k]=mean((predict(boost,newdata=train_data[-train_sample,],n.trees=t)
                   -train_data[-train_sample,]$Outcome)^2)
  }
}
mse
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,] 8.4269946 4.2337098 1.2191540 1.1780850 1.1269167
##  [2,] 2.4095949 0.6237138 0.2539578 0.2356993 0.2570798
##  [3,] 1.2383396 0.4409889 0.2510526 0.2489483 0.2675665
##  [4,] 0.8503159 0.3534733 0.2635092 0.2661129 0.2805647
##  [5,] 0.6217853 0.3514503 0.2664439 0.2680797 0.2866133
```

```
##  [6,] 0.5746594 0.3130677 0.2721715 0.2762933 0.2849708
##  [7,] 0.5217986 0.3522258 0.2804717 0.2798752 0.2732711
##  [8,] 0.4776513 0.3066790 0.2952511 0.2987943 0.2692242
##  [9,] 0.3924383 0.2981549 0.2760237 0.2876024 0.2850435
## [10,] 0.4882490 0.2878380 0.2733866 0.2840036 0.2789807
```

```
which.min(mse)
```

```
## [1] 32
```

```
min(mse)
```

```
## [1] 0.2356993
```

```
# minimized the validation set error for lambda=0.1 at n.trees=1000 and interaction.depth=2

# try a different random seed
set.seed(3)
train_sample=sample(1:nrow(train_data),0.7*nrow(train_data))
mse=matrix(rep(NA,5*10),nrow=10,ncol=5)
for (i in 1:10){
  k=0
  for (t in c(50,100,500,1000,5000)) {
    k=k+1
    boost=gbm(Outcome~.-Orientation,data=train_data[train_sample,],
                        distribution="gaussian",n.trees=t,interaction.depth=i,shrinkage=0.1)
    mse[i,k]=mean((predict(boost,newdata=train_data[-train_sample,],n.trees=t)
                   -train_data[-train_sample,]$Outcome)^2)
  }
}
mse
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,] 8.9543501 4.9886964 1.4455062 1.2614405 1.2593191
##  [2,] 2.5946840 0.7221849 0.3023429 0.2984264 0.2975001
##  [3,] 1.3954768 0.5469382 0.3290557 0.2890989 0.3002540
##  [4,] 0.9353197 0.3353352 0.3497049 0.3030225 0.3719850
##  [5,] 0.8204365 0.4068409 0.3716947 0.3364935 0.3643589
##  [6,] 0.5842915 0.3401215 0.3236788 0.3119265 0.3352088
##  [7,] 0.5723351 0.3557408 0.3798996 0.3541403 0.3472702
##  [8,] 0.6219760 0.4875575 0.3669895 0.3570441 0.4009663
##  [9,] 0.6389515 0.4189842 0.3848452 0.3881116 0.4088288
## [10,] 0.6311588 0.4177944 0.3526934 0.3501716 0.3916340
```

```
which.min(mse)
```

```
## [1] 33
```

```
min(mse)
```

```
## [1] 0.2890989
```

```
# minimized the validation set error for lambda=0.1 at n.trees=1000 and interaction.depth=3

# try a different random seed
set.seed(5)
train_sample=sample(1:nrow(train_data),0.7*nrow(train_data))
mse=matrix(rep(NA,5*10),nrow=10,ncol=5)
for (i in 1:10){
```

```
  k=0
  for (t in c(50,100,500,1000,5000)) {
    k=k+1
    boost=gbm(Outcome~.-Orientation,data=train_data[train_sample,],
                        distribution="gaussian",n.trees=t,interaction.depth=i,shrinkage=0.1)
    mse[i,k]=mean((predict(boost,newdata=train_data[-train_sample,],n.trees=t)
                   -train_data[-train_sample,]$Outcome)^2)
  }
}
mse
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,] 8.1074509 4.3454364 1.3434016 1.2688335 1.2465162
##  [2,] 2.3639329 0.5992894 0.2534306 0.2506944 0.2707106
##  [3,] 1.2516943 0.4704387 0.2691779 0.2846800 0.2786180
##  [4,] 0.9273757 0.3584599 0.2685800 0.2700234 0.2830648
##  [5,] 0.5834459 0.3133197 0.2885619 0.2947728 0.2879582
##  [6,] 0.5441769 0.3066708 0.2914068 0.2802074 0.2898056
##  [7,] 0.5235961 0.3083896 0.2866823 0.2991868 0.2877569
##  [8,] 0.5095344 0.3614866 0.2908736 0.2816820 0.3217168
##  [9,] 0.4485167 0.3046460 0.2845984 0.2974893 0.2947002
## [10,] 0.4525373 0.3208470 0.2831861 0.2987080 0.2949662
```

```
which.min(mse)
```

```
## [1] 32
```

```
min(mse)
```

```
## [1] 0.2506944
```

```
# minimized the validation set error for lambda=0.1 at n.trees=1000 and interaction.depth=2

# check lambda shrinkage values
new_train_data=data.frame(train_data,train_data$Wall.Area*train_data$Roof.Area)
set.seed(1)
train_sample=sample(1:nrow(train_data),0.7*nrow(train_data))
mse2=matrix(rep(NA,44*15),nrow=15,ncol=4)
for (i in 1:15){
  k=0
  for (j in c(0.001,0.01,0.1,0.5)) {
    k=k+1
    boost=gbm(Outcome~.-Orientation,data=new_train_data[train_sample,],
                        distribution="gaussian",n.trees=1000,interaction.depth=i,shrinkage=j)
    mse2[i,k]=mean((predict(boost,newdata=new_train_data[-train_sample,],n.trees=1000)
                    -new_train_data[-train_sample,]$Outcome)^2)
  }
}
mse2
```

```
##           [,1]      [,2]      [,3]      [,4]
##  [1,] 34.46458 4.5256084 1.1085598 1.1846886
##  [2,] 25.21744 0.6781660 0.2457018 0.2600060
##  [3,] 22.58064 0.4016347 0.2587512 0.2930131
##  [4,] 21.06402 0.3417410 0.2711878 0.2794352
##  [5,] 20.05605 0.3209140 0.2780032 0.3197044
##  [6,] 19.49148 0.2960732 0.2894040 0.3149741
```

```
##  [7,] 18.93208 0.2899752 0.2926055 0.3120116
##  [8,] 18.47293 0.2934875 0.2980758 0.3349989
##  [9,] 18.25697 0.2962993 0.3036552 0.2977612
## [10,] 18.00562 0.2944485 0.2972106 0.3547759
## [11,] 17.74055 0.3003499 0.2883307 0.3629244
## [12,] 17.60164 0.3020184 0.2789201 0.3802795
## [13,] 17.48431 0.2880404 0.2831540 0.3152795
## [14,] 17.46665 0.3008818 0.2682715 0.4388944
## [15,] 17.42828 0.3019342 0.2825170 0.3673021
```

```
which.min(mse2)
```

```
## [1] 32
```

```
min(mse2)
```

```
## [1] 0.2457018
```

```
# looks like validation set error is minimized for n.trees=1000 with
# interaction.depth=2 and shrinkage=0.1
# let's try this model

# fit to all training data and create test predictions
train_data=read.csv(file="/home/bridget/Dropbox/STATS202/kaggle/train.csv",header=T)
test_data=read.csv(file="/home/bridget/Dropbox/STATS202/kaggle/test.csv",header=T)
set.seed(1)
ID=test_data$ID
test_data=test_data[,-1]
test_data$Orientation=as.factor(test_data$Orientation)
test_data$Glazing.Distr=as.factor(test_data$Glazing.Distr)
train_data=train_data[,-1]
train_data$Orientation=as.factor(train_data$Orientation)
train_data$Glazing.Distr=as.factor(train_data$Glazing.Distr)
boost=gbm(Outcome~.-Orientation,data=train_data,
                      distribution="gaussian",n.trees=1000,interaction.depth=2,shrinkage=0.1)
out=data.frame(ID,predict(boost,newdata=test_data,n.trees=1000))
names(out)=c("ID","Outcome")
write.csv(out, file = "test_boost_rand_forest_bb_2017_11_28.csv",row.names=FALSE)
```