R Notebook

name: Saloni Mishra

Bagging Vs Boosting

```
#Data preparation
setwd("C://Users//salon//Desktop//DATA SCIENCE)

risk=read.csv("data.csv")

#Original CART model for predicting risk
choose=runif(dim(risk)[1],0,1)
train=risk[which(choose<=0.75),]
test=risk[which(choose>0.75),]
#View(test)
```

Original Cart Model for predicting risk

Accuracy of this model is very high i.e., 0.93

```
library(rpart)
cart.o=rpart(formula=risk~marital_status+mortgage+loans+income+age,
data=train, method="class")
p.0=predict(cart.o, newdata=test)
pred0=ifelse(p.0[,1]>p.0[,2],"Pred:bad loss", "Pred: good risk ")
o.t=table(pred0, test$risk)
o.t
```

```
##
## pred0              bad loss good risk
##   Pred: good risk         4        26
##   Pred:bad loss          33         2
```

Bagging Method*

Step 1 Samples (with replacement) are repeatedly taken from the training data set, so that each record has an equal probability of being selected, and each sample is the same size as the original training data set. These are the bootstrap samples.

Step 2 A classification or estimation model is trained on each bootstrap sample drawn in Step 1, and a prediction is recorded for each sample.

Step 3 The bagging ensemble prediction is then defined to be the class with the most votes in Step 2 (for classification models) or the average of the predictions made in Step 2 (for estimation models).

Accuracy of the original method bagging is same.

```
#Bagging Model
s1=train[sample(dim(train)[1], replace=TRUE),]
#s1

#Repeat the above for s2 through s5
cart1=rpart(risk~marital_status+mortgage+loans+income+age, data=s1,
method="class")
p1=predict(cart1, newdata=test)
pred1=ifelse(p1[,1]>p1[,2],"Pred: bad loss", "Pred: good risk")

s2 <- train[sample(dim(train)[1], replace = TRUE),]
cart2=rpart(risk~marital_status+mortgage+loans+income+age, data=s2,
method="class")
p2=predict(cart2, newdata=test)
pred2=ifelse(p2[,1]>p2[,2],"Pred: bad loss", "Pred: good risk")
############################################################
s3 <- train[sample(dim(train)[1], replace = TRUE),]
cart3=rpart(risk~marital_status+mortgage+loans+income+age, data=s3,
method="class")
p3=predict(cart3, newdata=test)
pred3=ifelse(p3[,1]>p3[,2],"Pred: bad loss", "Pred: good risk")
############################################################
s4 <- train[sample(dim(train)[1], replace = TRUE),]
cart4=rpart(risk~marital_status+mortgage+loans+income+age, data=s4,
method="class")
p4=predict(cart4, newdata=test)
pred4=ifelse(p4[,1]>p4[,2],"Pred: bad loss", "Pred: good risk")
############################################################
s5<-train[sample(dim(train)[1], replace=TRUE),]
cart5=rpart(risk~marital_status+mortgage+loans+income+age, data=s5,
method="class")
p5=predict(cart5, newdata=test)
```

```
pred5=ifelse(p5[,1]>p5[,2],"Pred: bad loss", "Pred: good risk")

#########################################################
preds=c(pred1, pred2, pred3, pred4, pred5)
recs=as.integer(names(preds));fin.pred=rep(0, dim(test)[1])
for(i in 1:dim(test)[1]){
  t = table(preds[which(recs==as.integer(rownames(test))[i])])
  fin.pred[i] = names(t)[t == max(t)]
}

bag.t=table(fin.pred, test$risk) # Contingency table
print(bag.t)

##
## fin.pred          bad loss good risk
##   Pred: bad loss        33         2
##   Pred: good risk        4        26
```

Boosting Method*

Step 1 All observations have equal weight in the original training data set . An initial "base" classifier is determined.

Step 2 The observations that were incorrectly classified by the previous base classifier have their weights increased, while the observations that were correctly classified have their weights decreased. This gives us data distribution . A new base classifier is determined, based on the new weights. This step is repeated until the desired number of iterations is achieved.

Step 3 The final boosted classifier is the weighted sum of the base classifiers.

Accuracy is 0.95

- Data Mining and Predictive Analytics: Larose

```
# Boosting model (5 iterations)
cart6 = rpart(risk ~ marital_status+mortgage+loans+income+age,
              data = train, method = "class")
p6 = predict(cart6, newdata = train)
pred6 = ifelse(p6[,1] > p6[,2], "bad loss", "good risk")
moreweight = train$risk != pred6
new.weights = ifelse(moreweight==TRUE, 2, 1)
cart7 = rpart(risk ~ marital_status+mortgage+loans+income+age,
              weights = new.weights, data = train, method = "class")

p7 = predict(cart7, newdata = train)
pred7 = ifelse(p7[,1] > p7[,2], "bad loss", "good risk")
moreweight = train$risk != pred7
new.weights = ifelse(moreweight==TRUE, 2, 1)
```

```r
cart8 = rpart(risk ~ marital_status+mortgage+loans+income+age,
              weights = new.weights, data = train, method = "class")

p8 = predict(cart8, newdata = train)
pred8 = ifelse(p8[,1] > p8[,2], "bad loss", "good risk")
moreweight = train$risk != pred8
new.weights = ifelse(moreweight==TRUE, 2, 1)

cart9 = rpart(risk ~ marital_status+mortgage+loans+income+age,
              weights = new.weights, data = train, method = "class")

p9 = predict(cart9, newdata = train)
pred9 = ifelse(p9[,1] > p9[,2], "bad loss", "good risk")
moreweight = train$risk != pred9
new.weights = ifelse(moreweight==TRUE, 2, 1)


# Repeat the above for cart8 and cart9
cart10 = rpart(risk ~ marital_status+mortgage+loans+income+age,
                weights = new.weights, data = train, method = "class")
p10 = predict(cart10, newdata = test)
pred10 = ifelse(p10[,1] > p10[,2], "Pred: bad loss", "Pred: good risk")
boost.t = table(pred10, test$risk) # Contingency table

# Compare contingency tables
o.t

##
## pred0              bad loss good risk
##   Pred: good risk         4        26
##   Pred:bad loss          33         2

bag.t

##
## fin.pred          bad loss good risk
##   Pred: bad loss        33         2
##   Pred: good risk        4        26

boost.t

##
## pred10            bad loss good risk
##   Pred: bad loss        33         2
##   Pred: good risk        4        26
```

Although it seems like as the accuracies are almost same models are alike but boosting has the least error as below

```
# Compare errors
(o.t[2]+o.t[3])/ sum(o.t)

## [1] 0.9076923

(bag.t[2]+bag.t[3])/ sum(bag.t)

## [1] 0.09230769

(boost.t[2]+boost.t[3])/ sum(boost.t)

## [1] 0.09230769
```

Conclusion: From the above analysis, we compared original CART method, Bagging and Boosting. The least error is of bagging.