HW6_Ronquillo

Melchor Ronquillo

2022-11-22

1. Describe the difference between bagging and boosting with trees.

```
Bagging creates multiple copies of original training data using the
bootstrap, fits a separate decision tree for each copy, combines all of
the trees in order to create one predictive model. Boosting does somewhat
the same but the trees grow sequentially, each tree grows using info from
previous trees. Boosting does not involve bootstrap sampling and instead,
each tree is fit on a modified version of the orignial data set
```

2. This question uses the Caravan data set (from the R package ISLR2).

```r
install.packages('ISLR2', repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/b4/vbzhgztj3tj299xgxnklp28c0000gn/T//Rtmp7JqdTK/downloaded_packages
```

```r
library(ISLR2)
car_data <- Caravan
#car_data

car_data$Purchase <- ifelse(car_data$Purchase =="Yes",1,0)
#car_data$Purchase <- as.factor(car_data$Purchase)
#car_data
```

```
(a) Create a training set consisting of the first 1,000 observations,
    and a test set consisting of the remaining observations.
```

```r
nrow(car_data)
```

```
## [1] 5822
```

```r
train_bst <- car_data[1:1000,]
test_bst <- car_data[1001:5822,]
```

```
(b) Fit a boosting model to the training set with Purchase as the response
    and the other variables as predictors. Use 1,000 trees, and a shrinkage
    value of 0.01. Which predictors appear to be the most important?
```

```r
install.packages('gbm', repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/b4/vbzhgztj3tj299xgxnklp28c0000gn/T//Rtmp7JqdTK/downloaded_packages
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
car_boost <- gbm(Purchase ~ ., data = train_bst, distribution = 'bernoulli', n.trees = 1000, shrinkage =
```

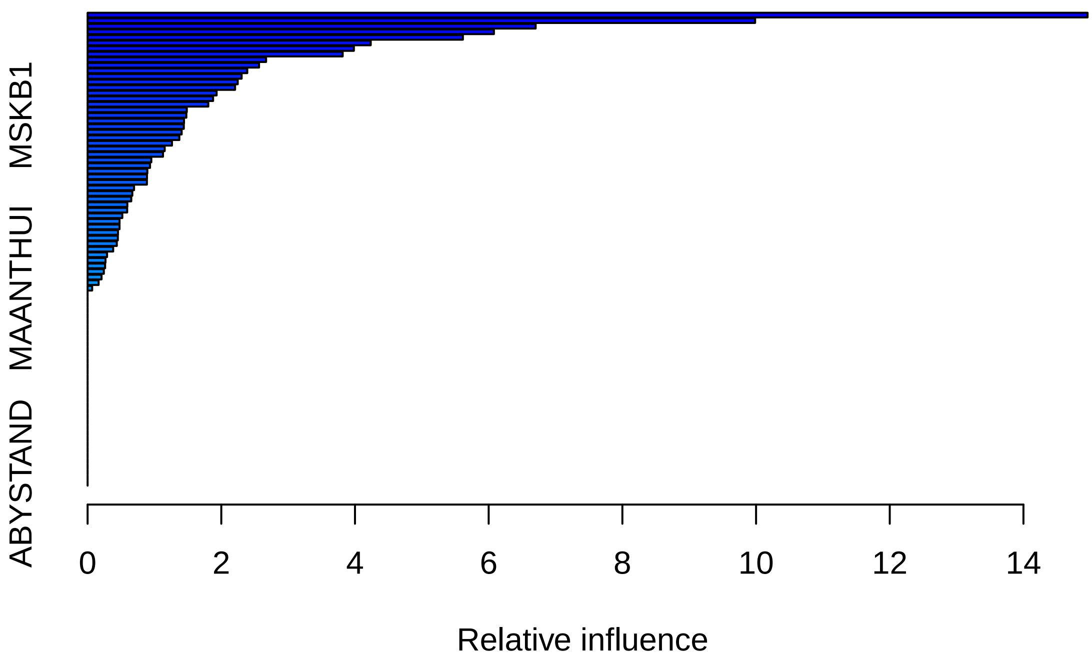
```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 50: PVRAAUT has no variation.

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

```
car_boost
```

```
## gbm(formula = Purchase ~ ., distribution = "bernoulli", data = train_bst,
##     n.trees = 1000, shrinkage = 0.01)
## A gradient boosted model with bernoulli loss function.
## 1000 iterations were performed.
## There were 85 predictors of which 50 had non-zero influence.
```

```
summary(car_boost)
```




```
##                   var     rel.inf
## PPERSAUT PPERSAUT 14.95946391
## MKOOPKLA MKOOPKLA  9.98599184
## MOPLHOOG MOPLHOOG  6.70296721
## MBERMIDD MBERMIDD  6.07825606
## PBRAND     PBRAND  5.61432185
## MGODGE     MGODGE  4.23556162
## ABRAND     ABRAND  3.98371773
## MINK3045 MINK3045  3.81584367
## MAUT2       MAUT2  2.67008309
## MSKA         MSKA  2.56495557
## MSKC         MSKC  2.38861218
## MOSTYPE   MOSTYPE  2.30414832
## MGODPR     MGODPR  2.24603833
## MBERARBG MBERARBG  2.20613809
## PWAPART   PWAPART  1.92996992
## MINKGEM   MINKGEM  1.87757082
```

```
## MAUT1       MAUT1  1.80581902
## MGODOV     MGODOV  1.48390583
## MSKB1       MSKB1  1.47713413
## MRELGE     MRELGE  1.44204552
## PBYSTAND PBYSTAND  1.43948579
## MINKM30   MINKM30  1.40719946
## MBERHOOG MBERHOOG  1.37238441
## MHHUUR     MHHUUR  1.26209112
## MRELOV     MRELOV  1.15331098
## MFWEKIND MFWEKIND  1.12772743
## MOPLMIDD MOPLMIDD  0.95294717
## MFGEKIND MFGEKIND  0.93397334
## MINK7512 MINK7512  0.89345648
## MAUT0       MAUT0  0.88836165
## MGEMLEEF MGEMLEEF  0.88712875
## PMOTSCO   PMOTSCO  0.69490563
## MBERBOER MBERBOER  0.66899492
## MGODRK     MGODRK  0.65304866
## MINK4575 MINK4575  0.59454253
## MOSHOOFD MOSHOOFD  0.59186374
## MHKOOP     MHKOOP  0.51891214
## MSKB2       MSKB2  0.47788670
## PLEVEN     PLEVEN  0.47745189
## MZFONDS   MZFONDS  0.45459699
## MSKD         MSKD  0.45231884
## MINK123M MINK123M  0.43685057
## MGEMOMV   MGEMOMV  0.38264877
## MOPLLAAG MOPLLAAG  0.29123598
## MBERARBO MBERARBO  0.26776213
## MZPART     MZPART  0.26195749
## APERSAUT APERSAUT  0.24276853
## MFALLEEN MFALLEEN  0.20855101
## MRELSA     MRELSA  0.16496313
## MAANTHUI MAANTHUI  0.06812908
## MBERZELF MBERZELF  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
```

```
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## ALEVEN     ALEVEN  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

Out of the 85 total predictors, 50 of them had non-zero influence. Predictors such has PPERSUAT, MKOOPKLA, MOPLHOOG, MBERMIDD, PBRAND, ABRAND, MGODGE, and MINK3045 all had relative influence greater than 3, while many others being between 0 - 2.

(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20%. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

```
set.seed(1111)
yhat_boost <- predict(car_boost, newdata = test_bst, type = "response")
```

```
## Using 1000 trees...
```

```
#yhat_boost
test_predict <- ifelse(yhat_boost > 0.2, 1, 0)
#test_predict
table(test_predict, test_bst$Purchase)
```

```
##
## test_predict    0    1
##            0 4419  256
##            1  114   33
```

Out of the 165 people predicted to make a purchase, only 36 people actually made one (28% accuracy).

NOTE: the predictions and table values are always changing after each knit even with a seed, however the accuracies are always around the same

```
set.seed(2)
car_log <- glm(Purchase ~ ., data = train_bst, family = "binomial")
```

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

```r
#car_log
yhat_log <- predict(car_log, newdata = test_bst, type = "response")
```

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

```r
#yhat_log
log_predict <- ifelse(yhat_log> 0.2, 1, 0)
table(log_predict, test_bst$Purchase)
```

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

For logistic regression, out of the 408 people predicted to make a purchase, only 58 actually did (14% accuracy)

```r
install.packages('class', repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/b4/vbzhgztj3tj299xgxnklp28c0000gn/T//Rtmp7JqdTK/downloaded_packages
```

```r
library(class)
set.seed(3)
car_knn <- knn(train=train_bst, test=test_bst, cl=train_bst$Purchase, k=1)
#car_knn
table(car_knn ,test_bst$Purchase)
```

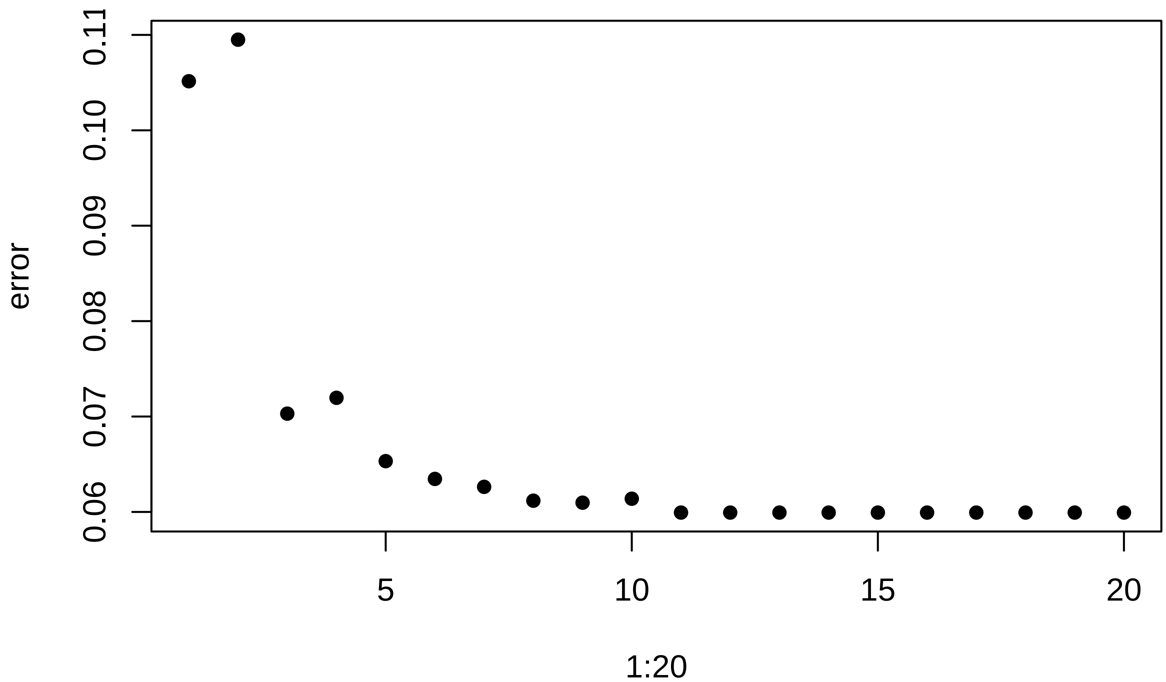
```
##
## car_knn    0    1
##       0 4284  262
##       1  249   27
```

For knn with k = 1, out of the 276 people predicted to make a purchase, 27 people did (9% accuracy)

```r
set.seed(4)
error <- c()
for (k in 1:20){#print(k)
car_knncv <- knn(train=train_bst, test=test_bst, cl=train_bst$Purchase, k=k)

#table(holdout$col, greg)
error[k] <- mean(test_bst$Purchase != car_knncv)
}

plot(1:20, error, pch = 16)
```




```
set.seed(5)
car_knn2 <- knn(train=train_bst, test=test_bst, cl=train_bst$Purchase, k=8)
#car_knn2
table(car_knn2 ,test_bst$Purchase)
```

```
##
## car_knn2    0    1
##        0 4525  287
##        1    8    2
```

```
Using cross validation to find the best k, with k = 8, out of 10
people predicted to make a purchase only 2 did (20%)

Overall I believe that all these classifiers did a good job
predicting who will not make a purchase and actually did not make
a purchase, and although the accuracies were not that great in
predcicting who would make a purchase that actually made a purchase
the boosting model had the best accuracy in comparison to logistic
and KNN.
```