HW5 Responses

Yawei LI

2/27/2020

Question 1

When growing a classification tree, Gini index and the cross-entropy are expected to do a better job than plain error rate because they react faster to impurity in classification.

When pruning a decision tree, classification error will come in handy because they are closer to the very concept of "correct classification" that we try to achieve.

Question 2

Data and Prep

```
train = read.csv('gss_train.csv')
test = read.csv('gss_test.csv')
train$colrac = as.factor(train$colrac)
test$colrac = as.factor(test$colrac)
models = c("Logistic regression",
'Naive Bayes',
'Elastic net regression',
'Decision tree (CART)',
'Bagging',
'Random forest',
'Boosting')
cv_error = numeric()
auc = numeric()
```

Logistic

```
set.seed(1234)
train.control <- trainControl(method = "cv", number = 10)
logit <- train(colrac ~., data = train, method = "glm", family = 'binomial',
               trControl = train.control)
cv_error[1] <- logit$results$Accuracy
logit_pred <- as.numeric(predict(logit,newdata = train))
auc[1] <- roc(train$colrac, logit_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

naive Bayes

```r
set.seed(2234)
train.control <- trainControl(method = "cv", number = 10)
nb <- train(colrac ~., data = train, method = "nb",
               trControl = train.control)
cv_error[2] <- nb$results$Accuracy
#I got 2 accuracies here labeled for T and F, not sure which to use.
nb_pred <- as.numeric(predict(nb,newdata = train))
auc[2] <- roc(train$colrac, nb_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

Elastic Net

```r
set.seed(3234)
train.control <- trainControl(method = "cv", number = 10)
enet <- train(colrac ~., data = train, method = "glmnet",
               trControl = train.control)
cv_error[3] <- enet$results$Accuracy[5] # Chosen by reading summary
enet_pred <- as.numeric(predict(enet,newdata = train))
auc[3] <- roc(train$colrac, enet_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

CART

```r
set.seed(4234)
cart <- tree(colrac ~., data = train)
cart_cv <- cv.tree(cart, FUN = prune.misclass)
min_idx <- which.min(cart_cv$dev)
best_size <- cart_cv$size[min_idx]
cart_prune <- prune.misclass(cart, best = best_size)
cv_error[4] <- 1 -summary(cart_prune)$misclass[1]/summary(cart_prune)$misclass[2]
# Misclassifications / Total number of obs.
cart_pred <- as.numeric(predict(cart_prune,newdata = train, type = 'class'))
auc[4] <- roc(train$colrac, cart_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

Bagging

```r
set.seed(5234)
ctrl <- trainControl(method = "cv",  number = 10)
bagged <- train(
  colrac ~ .,
```

```r
  data = train,
  method = "treebag",
  trControl = ctrl
)
cv_error[5] <- bagged$results$Accuracy
bagged_pred <- as.numeric(predict(bagged,newdata = train))
auc[5] <- roc(train$colrac, bagged_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Random Forest

```r
set.seed(6234)
rf1 <- train(
  colrac ~ .,
  data = train,
  method = "ranger",
  trControl = ctrl,
  # splitrule = 'gini' # This is also causing error same as below.
  importance = "impurity"
) # Maxi Accuracy 0.801 at 3rd model


# I am tring to tuning node size here.
# But encountered an error of
# formal argument min.node.size matched by multiple actual arguments
# by adding min.node.size = 3, default is 1.
# The caret documentation said it could tune min.node.side, yet
# I got it set at default.

#set.seed(7234)
#rf1 <- train(
#  colrac ~ .,
#  data = train,
#  method = "ranger",
#  trControl = ctrl,
#  importance = "impurity",
#  min.node.size = 3
#)

cv_error[6] <- rf1$results$Accuracy[3]
rf_pred <- as.numeric(predict(rf1,newdata = train))
auc[6] <- roc(train$colrac, rf_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Boosting

```
set.seed(7234)
boost <- train(
  colrac ~ .,
  data = train,
  method = "gbm",
  trControl = trainControl(method = "cv",
                           number = 10,
                           verboseIter = FALSE),
  verbose = 0)
cv_error[7] <- boost$results$Accuracy[8] # Max accuracy 0.809 at 8th model
boost_pred <- as.numeric(predict(boost,newdata = train))
auc[7] <- roc(train$colrac, boost_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

Question 3 & 4 Model Comparision

As can be seen from the table below, Boosting model provides the best cross-validation correct classification rate, while maintains a level of area under curve (the best if we ignore the seemingly problematic bagging and random forest auc). Therefore, I believe boosting model performs the best according to our rubics.

```
report <- data.frame(models, cv_error, auc)
print(report)
```

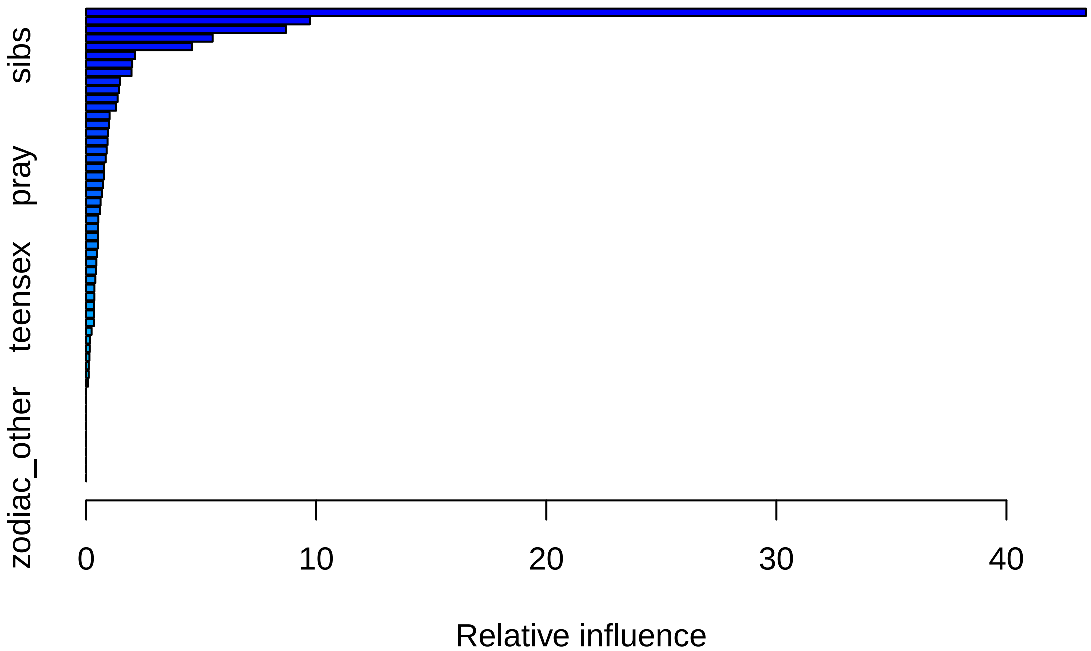
```
##                    models  cv_error       auc
## 1     Logistic regression 0.7994346 0.8164760
## 2             Naive Bayes 0.7371760 0.7431246
## 3 Elastic net regression 0.8021925 0.8179706
## 4    Decision tree (CART) 0.7621951 0.7535254
## 5                 Bagging 0.7784657 0.9972150
## 6           Random forest 0.8001178 1.0000000
## 7                Boosting 0.8089125 0.8505333
```

As can be seen from the feature importance plot and table below. A 44 feature model may seem over complicated to a certain point, yet a closer examination reveals that the model does drop quite a number of insignificant features and assigned very different weights to important and unimportant features. Therefore, though it may suffer from a certain degree of over-fitting, it also takes better advantage of the large number of features in our gss data, and limits the effects of over fitting to a smaller degree.

```
print(boost$finalModel)
```

```
## A gradient boosted model with bernoulli loss function.
## 150 iterations were performed.
## There were 55 predictors of which 44 had non-zero influence.
```

```
print(summary(boost$finalModel))
```




```
##                                             var     rel.inf
## tolerance                             tolerance 43.46272149
## colmslm                                 colmslm  9.71902432
## colath                                   colath  8.68012869
## colmil                                   colmil  5.49602033
## age                                         age  4.60528792
## sibs                                       sibs  2.12984569
## income06                               income06  2.00327861
## science_quiz                       science_quiz  1.97087363
## social_connect                   social_connect  1.47990249
## egalit_scale                       egalit_scale  1.41783459
## polviews                               polviews  1.36620771
## wordsum                                 wordsum  1.30502040
## con_govt                               con_govt  1.01723045
## authoritarianism               authoritarianism  1.00017260
## tvhours                                 tvhours  0.94219244
## mode                                       mode  0.92940211
## colcom                                   colcom  0.88987921
## homosex                                 homosex  0.84722764
## marital_Never.married   marital_Never.married  0.78418141
## pray                                       pray  0.76484468
## south                                     south  0.72516587
## social_cons3_Conserv     social_cons3_Conserv  0.69252993
## childs                                   childs  0.62554991
## grass                                     grass  0.60915174
## colhomo                                 colhomo  0.52800977
```

```
## happy                                          happy  0.52345096
## attend                                        attend  0.52250550
## evangelical                              evangelical  0.50387497
## owngun                                        owngun  0.46616147
## marital_other                          marital_other  0.44275076
## degree_other                            degree_other  0.41860186
## partyid_3_Ind                          partyid_3_Ind  0.39904746
## spend3_Mod                                spend3_Mod  0.35734474
## teensex                                      teensex  0.35639610
## vetyears                                    vetyears  0.34332031
## partyid_3_Rep                          partyid_3_Rep  0.33451337
## marital_Divorced                    marital_Divorced  0.33171433
## reborn_r                                    reborn_r  0.23580795
## relig_other                              relig_other  0.17463947
## spend3_Liberal                        spend3_Liberal  0.15264655
## hispanic_2                                hispanic_2  0.14093224
## pres08                                        pres08  0.11064928
## sex                                              sex  0.10894407
## pornlaw2                                    pornlaw2  0.08501498
## black                                          black  0.00000000
## born                                            born  0.00000000
## degree_Bachelor.deg              degree_Bachelor.deg  0.00000000
## news_FEW.TIMES.A.WEEK          news_FEW.TIMES.A.WEEK  0.00000000
## news_LESS.THAN.ONCE.WK news_LESS.THAN.ONCE.WK  0.00000000
## news_NEVER                                news_NEVER  0.00000000
## news_other                                news_other  0.00000000
## relig_CATHOLIC                        relig_CATHOLIC  0.00000000
## relig_NONE                                relig_NONE  0.00000000
## social_cons3_Mod                    social_cons3_Mod  0.00000000
## zodiac_other                            zodiac_other  0.00000000
```

Question 5 Apply to Test

As can be seen, our best model did a mediocre job on the test set, yet is not much a decrease compared to the training set. This means the overfitting problem is handled at least partically, yet the overall prediction accruacy could use more work.

```
boost_test_pred <- as.numeric(predict(boost,newdata = test))
error_rate <- sum(boost_test_pred - 1 == test$colrac)/nrow(test)
auc_test <- roc(test$colrac, boost_test_pred)$auc[1]
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
paste('correct classification rate is', error_rate)
```

```
## [1] "correct classification rate is 0.799188640973631"
```

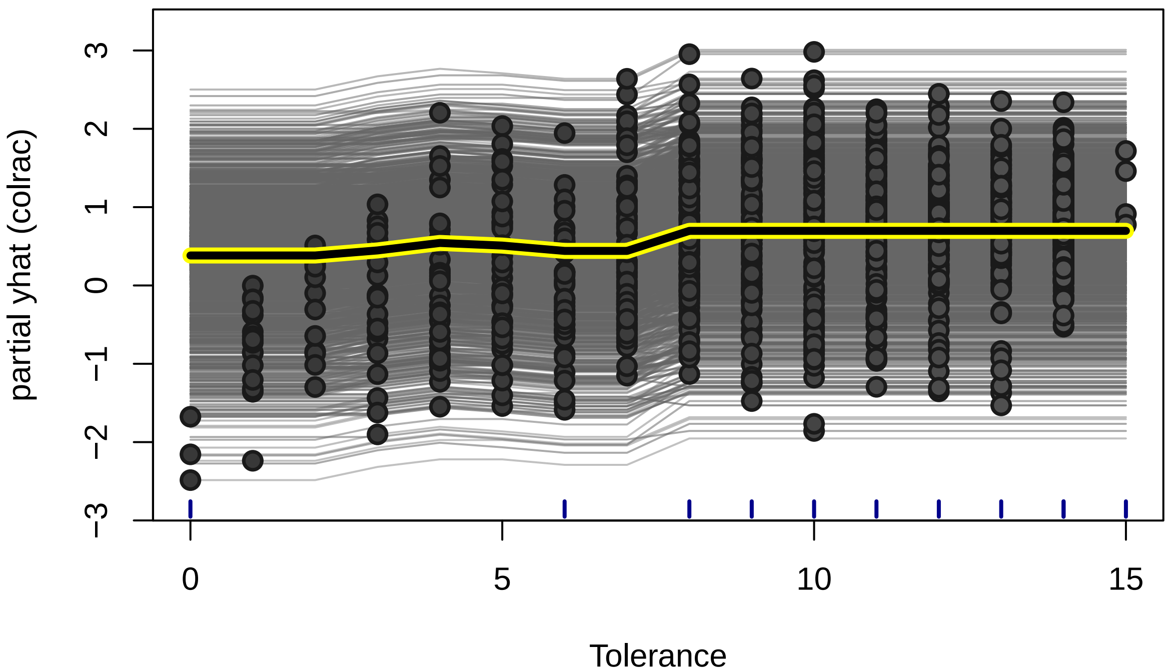
```
paste('area under curve is', auc_test)
```

```
## [1] "area under curve is 0.792999006951341"
```

Question 6 Interpretation with ICE curve

For tolerance:

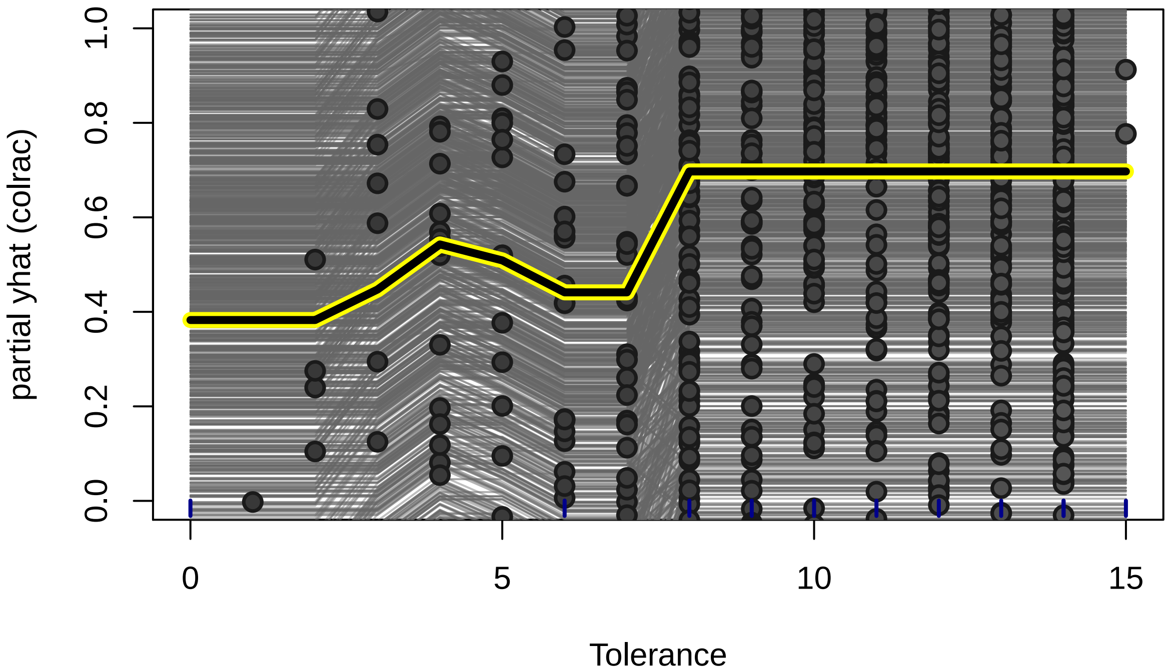
```
train = read.csv('gss_train.csv')
train$colrac = as.numeric(train$colrac) - 1
#levels(train$colrac) <- c('1','2') # Failed attempt to solve duplicate level problem

# Reloading train to reverse the type change to $colrac that is causing errors
# by 'duplicated levels'. The code below treat our colrac as numeric.

ice1 <- ice(
  object = boost$finalModel,
  X = train,
  y = train$colrac,
  predictor = which(colnames(train) == 'tolerance'),
  verbose = FALSE,
  #logodds = TRUE, This should be included for classification,
  # but I'm not doing it for the prolem metioned above.
  n.trees = 150
  # An arbitrary large value because the number of trees
  # used cannot be found at boost$finalmodel descriptions
  )

plot(ice1,
     xlab = 'Tolerance',
     ylab = 'partial yhat (colrac)')
```



As can be seen from the plot, the level of tolerance has a positive effect on predicted outcome of whether allowing racist to teach at school. As the tolerance level increases, the yhat also increases unstably until tolerance level reaches 8. After 8, the effect of tolerance increase does not seem to have any further impact on the dependant variable.

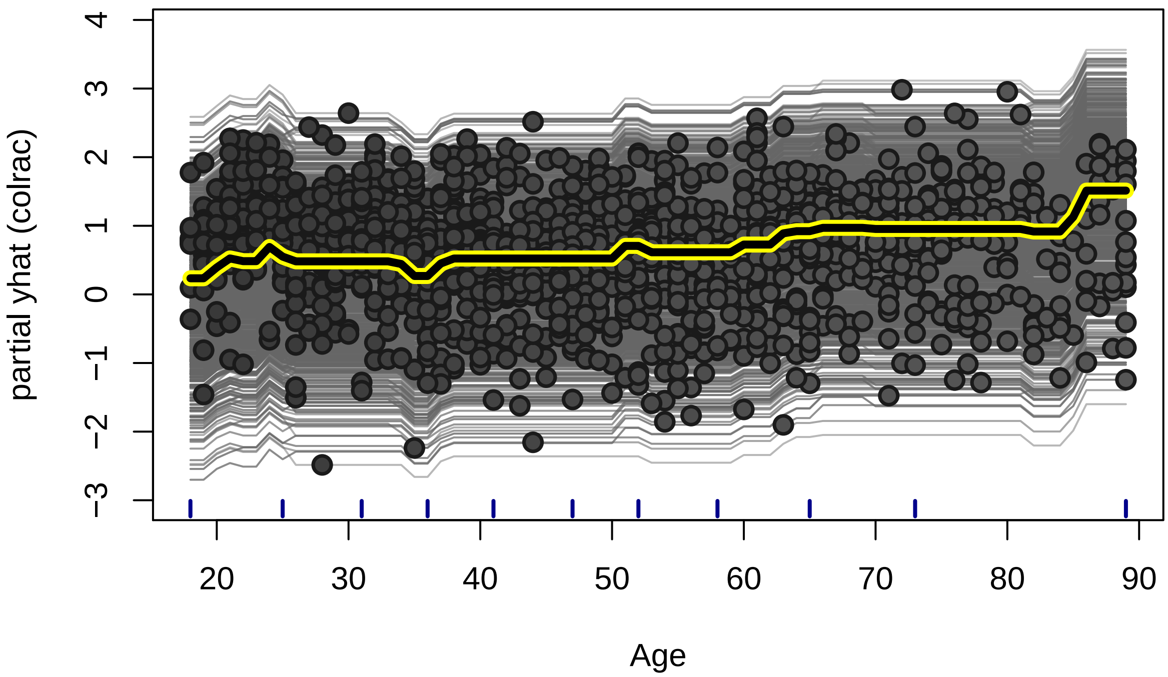
Below is the same plot with shallower ylim to further illustrate the conditional effect.

```
plot(ice1,
     xlab = 'Tolerance',
     ylab = 'partial yhat (colrac)',
     ylim = c(0,1)
)
```



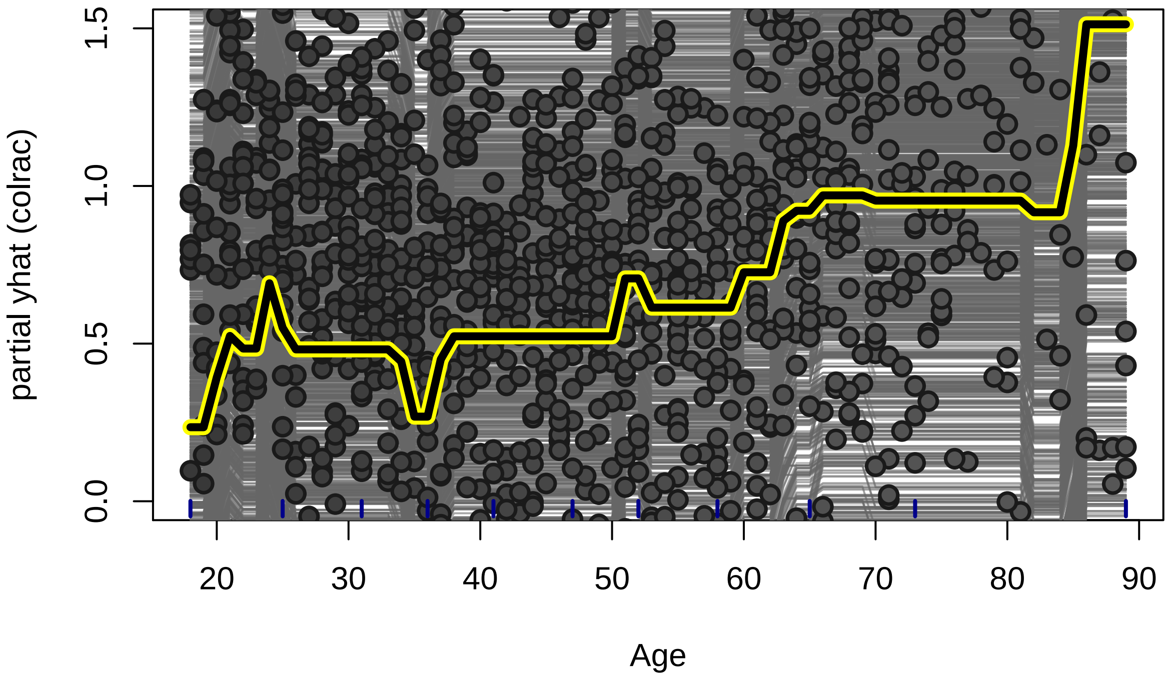
For age:

```
ice2 <- ice(
  object = boost$finalModel,
  X = train,
  y = train$colrac,
  predictor = 'age',
  verbose = FALSE,
  #logodds = TRUE, This should be included for classification,
  # but I'm not doing it for the prolem metioned above.
  n.trees = 150)

plot(ice2,
     xlab = 'Age',
     ylab = 'partial yhat (colrac)')
```



Zoomed-in version:

```
plot(ice2,
     xlab = 'Age',
     ylab = 'partial yhat (colrac)',
     ylim = c(0,1.5)
)
```



As can be seen from the plot, age on average has a positive effect on whether let racists teachers stay. The effect is rather unstable and weak at before 60 years old and we've seen a drop on mid 30s (where most parents have their children at school age). After 60 years old the effect becomes strong and stable and there's a spike at 80+.