PS 5 Borui Sun

borui sun

2/29/2020

Conceptual: Cost functions for classification trees

1. (15 points) Consider the Gini index, classification error, and cross-entropy in simple classification settings with two classes. Of these three possible cost functions, which would be best to use when growing a decision tree? Which would be best to use when pruning a decision tree? Why?

Either Gini index or cross-entropy would be the best to use when growing a decision tree because both functions are sensitive to changes in the node probability and favor pure nodes. Mathematically, Gini index will be minimized when all records belong to a single class, and similarly cross-entropy will be minimized when one class has no records in it. However, node purity does not affect classification error if the accuracy does not improve.

Classification error would be the best to use when pruning a decision tree. Because when pruning the tree, we want to preserve the branches and nodes that maximize the accuracy.

Application: Predicting attitudes towards racist college professors

2. (35 points; 5 points/model) Estimate the following models, predicting `colrac` using the training set (the training `.csv`) with 10-fold CV:

   - Logistic regression
   - Naive Bayes
   - Elastic net regression
   - Decision tree (CART)
   - Bagging
   - Random forest
   - Boosting

```r
# Load data
gss_train <- read_csv("./data/gss_train.csv")
y.train <- gss_train$colrac %>% as.factor()
x.train <- gss_train %>% select(-colrac) %>% mutate_all(~as.vector(scale(.)))

gss_test <- read_csv("./data/gss_test.csv")
y.test <- gss_test$colrac %>% as.factor()
x.test <- gss_test %>% select(-colrac) %>% mutate_all(~as.vector(scale(.)))

set.seed(233)
ctrl <- trainControl(method = "cv", number = 10)

# Logistic
Logistic.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "glm",
  family = "binomial"
)

# Naive Bayes
Bayes.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "nb")


# Elastic Net
Elnet.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "glmnet",
  tuneLength = 100
)

# Decision Tree (CART)
Cart.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "rpart",
  tuneLength = 100
)

# Bagging
Bagging.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "treebag"
)

# Random Forest
rf.grid <- expand.grid(.mtry =seq(2, ncol(gss_train) - 1, by = 5))

Rf.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "rf",
  tuneGrid = rf.grid
)

# Boosting
bosting.grid <- expand.grid(.n.trees = seq(500, 2000, by = 500),
                            .interaction.depth = c(1:4),
                            .shrinkage = seq(0.0001, 0.04, by = 0.001),
                            .n.minobsinnode = seq(5, 30, by = 5))
```

```
Boosting.fit <- train(
  x.train,
  y.train,
  trControl = ctrl,
  method = "gbm"
)
```

Evaluate the models

3. (20 points) Compare and present each model's (training) performance based on

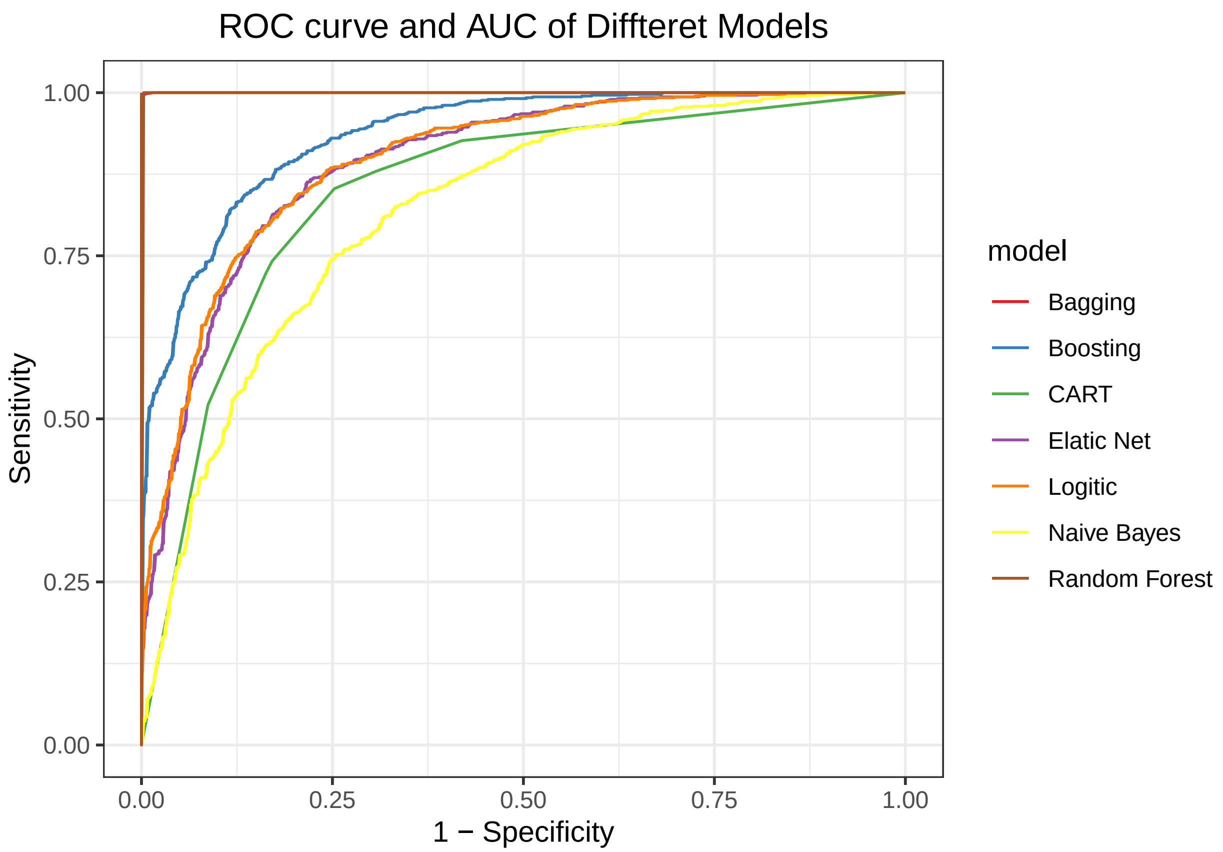
   - Cross-validated error rate
   - ROC/AUC

```
# Get error rate and AUC

# Logistic
logis.er <- Logistic.fit$results$Accuracy
logis.rocinfo <- roc(gss_train$colrac, predict(Logistic.fit, x.train, type = "prob")[,2])
logis.auc <- logis.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Logitic",
                        tp = logis.rocinfo$sensitivities,
                        fp = 1 - logis.rocinfo$specificities)


# Naive Bayes
nb.er <- Bayes.fit$results %>% filter(fL == Bayes.fit$bestTune$fL,
                                      usekernel == Bayes.fit$bestTune$usekernel,
                                      adjust == Bayes.fit$bestTune$adjust) %>% pull(Accuracy)
nb.rocinfo <- roc(gss_train$colrac, predict(Bayes.fit, x.train, type = "prob")[,2])
nb.auc <- nb.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Naive Bayes",
                        tp = nb.rocinfo$sensitivities,
                        fp = 1 - nb.rocinfo$specificities) %>% bind_rows(roc_plot_prep)


# Elastic Net
elnet.er <- Elnet.fit$results %>% filter(alpha == Elnet.fit$bestTune$alpha,
                                         lambda == Elnet.fit$bestTune$lambda) %>% pull(Accuracy)
elnet.rocinfo <- roc(gss_train$colrac, predict(Elnet.fit, x.train, type = "prob")[,2])
elnet.auc <- elnet.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Elatic Net",
                        tp = elnet.rocinfo$sensitivities,
                        fp = 1 - elnet.rocinfo$specificities) %>% bind_rows(roc_plot_prep)


# CART
cart.er <- Cart.fit$results %>% filter(cp == Cart.fit$bestTune$cp) %>% pull(Accuracy)
cart.rocinfo <- roc(gss_train$colrac, predict(Cart.fit, x.train, type = "prob")[,2])
cart.auc <- cart.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "CART",
                        tp = cart.rocinfo$sensitivities,
                        fp = 1 - cart.rocinfo$specificities) %>% bind_rows(roc_plot_prep)
```

```r
# Bagging
bagging.er <- Bagging.fit$results$Accuracy
bagging.rocinfo <- roc(gss_train$colrac, predict(Bagging.fit, x.train, type = "prob")[,2])
bagging.auc <- bagging.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Bagging",
                        tp = bagging.rocinfo$sensitivities,
                        fp = 1 - bagging.rocinfo$specificities) %>% bind_rows(roc_plot_prep)


# Random Forest
rf.er <- Rf.fit$results %>% filter(mtry == Rf.fit$bestTune$mtry) %>% pull(Accuracy)
rf.rocinfo <- roc(gss_train$colrac, predict(Rf.fit, x.train, type = "prob")[,2])
rf.auc <- rf.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Random Forest",
                        tp = rf.rocinfo$sensitivities,
                        fp = 1 - rf.rocinfo$specificities) %>% bind_rows(roc_plot_prep)

# Boosting
boosting.er <- Boosting.fit$results %>% filter(
  n.trees == Boosting.fit$bestTune$n.trees,
  interaction.depth == Boosting.fit$bestTune$interaction.depth,
  shrinkage == Boosting.fit$bestTune$shrinkage,
  n.minobsinnode == Boosting.fit$bestTune$n.minobsinnode) %>% pull(Accuracy)
boosting.rocinfo <- roc(gss_train$colrac, predict(Boosting.fit, x.train, type = "prob")[,2])
boosting.auc <- boosting.rocinfo$auc %>% as.numeric()
roc_plot_prep <- tibble(model = "Boosting",
                        tp = boosting.rocinfo$sensitivities,
                        fp = 1 - boosting.rocinfo$specificities) %>% bind_rows(roc_plot_prep)


tibble(
  model = c("Logistic", "Naive Bayes", "Elastic Net", "CART", "Bagging", "Random Forest", "Boosting"),
  CV.error.rate = c(logis.er, nb.er, elnet.er, cart.er, bagging.er, rf.er, boosting.er),
  AUC = c(logis.auc, nb.auc, elnet.auc, cart.auc, bagging.auc, rf.auc, boosting.auc)) %>%
  mutate(CV.error.rate = 1- CV.error.rate) %>% kable()
```

model	CV.error.rate	AUC
Logistic	0.2025878	0.8965404
Naive Bayes	0.2602426	0.8164134
Elastic Net	0.1910707	0.8929971
CART	0.2127843	0.8487654
Bagging	0.2114104	0.9999761
Random Forest	0.1883848	1.0000000
Boosting	0.1856637	0.9158713

```r
roc_plot_prep %>%
  ggplot(aes(x = fp, y = tp, color = model)) +
  geom_line() +
  scale_color_brewer(palette = "Set1") +
  labs(title = "ROC curve and AUC of Diffteret Models",
       x = "1 - Specificity", y ="Sensitivity")
```



4. (15 points) Which is the best model? Defend your choice.

Random Forest

According to the table and ROC plot above, there are four models that perform relatively better than the others with respects to CV error rate and AUC. The four models are elastic net, bagging, random forest and boosting. I choose random forest as the best model over the others for the following reasons:

1. Between bagging and random forest: The fundamental difference between the two models is that random forest only uses a subset of features while bagging uses all of them when splitting a node. If we look at the best-tuned random forest model, 22 of 55 features are used. Random forest further reduces variance and hence is less prone to overfitting than bagging. Random forest also has a much smaller CV error rate. Therefore, I rule out bagging from my selection.

2. Between elastic net and random forest: While elastic net has the best performance among parametric models, its CV error rate is higher than the other two tree-based models and AUC is lower than them. Considering that it is unlikely that the underlying relationship is linear, I also rule out elastic net as well as other linear models.

3. Between boosting and random forest: It is really difficult to choose between the two models intuitively. Boosting builds on weaker learners and gradually improves them. Random Forest starts with a big tree and then pruns it. If we look at their best-tuned model, neither one of them seems too complicated. Random forest only uses 22 features and boosting has 100 trees. However, the CV error rate of boosting is only marginally smaller than the CV error rate of random forest (~0.003), but random forest has a much greater AUC than boosting (~0.1). Therefore, I believe that random forest will perform better.

```
# Parameters of random forest
Rf.fit$bestTune %>% kable()
```

	mtry
5	22

```
# Parameters of boosting
Boosting.fit$bestTune %>% kable()
```

	n.trees	interaction.depth	shrinkage	n.minobsinnode
5	100	2	0.1	10

Evaluate the *best* model

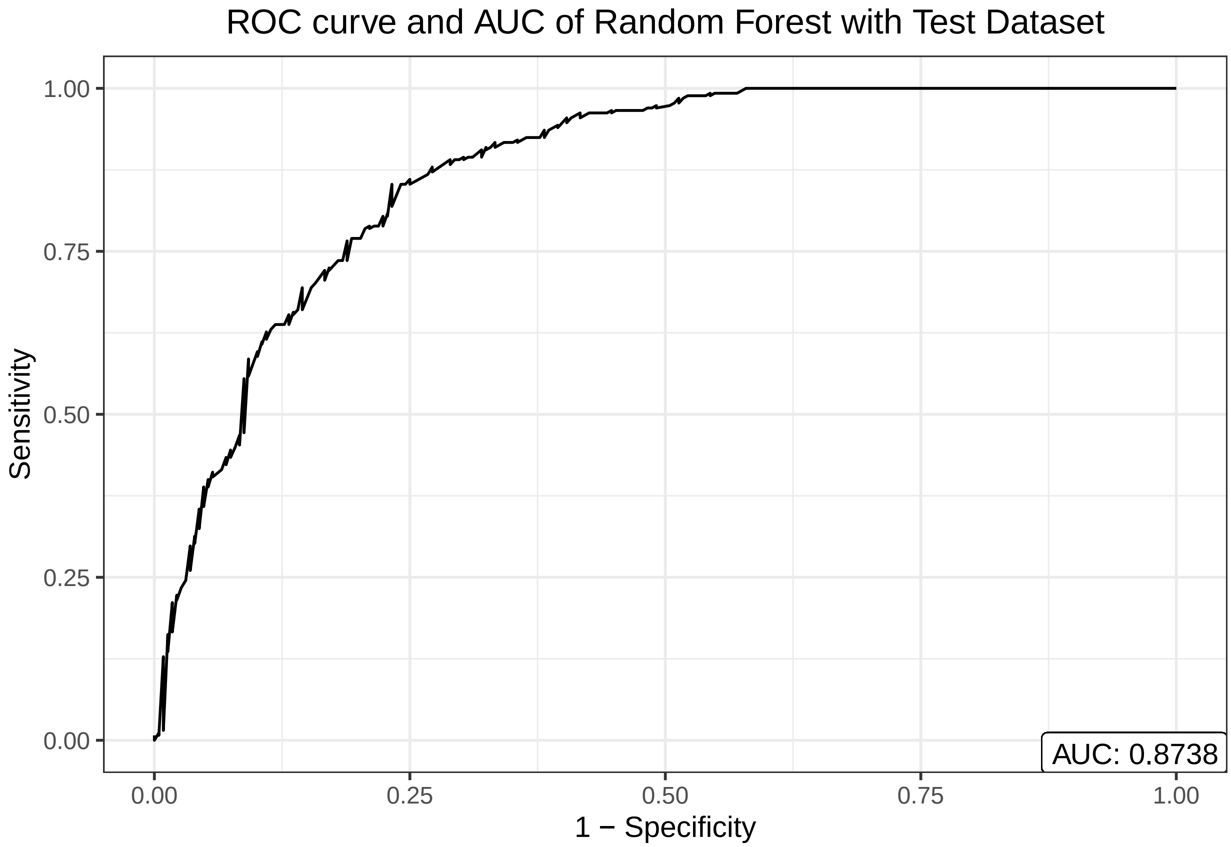
5. (15 points) Evaluate the *final*, best model's (selected in 4) performance on the test set (the test `.csv`) by calculating and presenting the classification error rate and AUC. Compared to the fit evaluated on the training set in questions 3-4, does the "best" model generalize well? Why or why not? How do you know?

I think my "best" model - random forest- generalize quite well. Although the test error rate is higher the train error rate (~0.12), an accuracy close to 80% is still quite high. The increase in test error rate is expected, as the test error rate is always larger than the training error rate due to irreducible error. The AUC plot of the "best" model below shows a AUC of 0.87. Although the test AUC is smaller than the train, it is definitely not a sign of bad performance. In our training data, random forest has a AUC of 1, which is extremely high and probably due to overfitting. An AUC of 0.87 is still very high.

```
cat("Classification Error Rate:",
    predict(Rf.fit, x.test) %>% {mean(. != y.test)})
```

```
## Classification Error Rate: 0.2028398
```

```
rocinfo <- roc(gss_test$colrac, predict(Rf.fit, x.test, type = "prob")[,2])

tibble(tp = rocinfo$sensitivities,
       fp = 1 - rocinfo$specificities,
       auc = as.numeric(rocinfo$auc)) %>%
  ggplot(aes(x = fp, y = tp)) +
  geom_line() +
  geom_label(aes(label = paste("AUC:", round(auc,4))), x = Inf, y = -Inf, hjust = 1, vjust = 0) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "ROC curve and AUC of Random Forest with Test Dataset",
        x = "1 - Specificity", y ="Sensitivity")
```



More importantly, even though random forest performs relatively worse on the test dataset than the training dataset, it is very likely that other models may also perform worse on test dataset. Even if there is one model that beats random forest on the test dataset, the discrepancy on error rate and AUC would not be large. To demonstrate this idea, I also calculate the error rates and AUCs from the other three models I picked out in Q.4. As shown below, random forest has the highest AUC and the second lowest error rate.

```
cat("Classification Error Rate:", "\n",
    "Elastic Net:",
    predict(Elnet.fit, x.test) %>% {mean(. != y.test)}, "\n",
    "Bagging:",
    predict(Bagging.fit, x.test) %>% {mean(. != y.test)},"\n",
    "Boosting:",
    predict(Boosting.fit,  x.test) %>% {mean(. != y.test)}, "\n",
    "AUC:", "\n",
    "Elastic Net",
    roc(gss_test$colrac, predict(Elnet.fit, x.test, type = "prob")[,2])$auc, "\n",
    "Bagging",
    roc(gss_test$colrac, predict(Bagging.fit, x.test, type = "prob")[,2])$auc, "\n",
    "Boosting",
    roc(gss_test$colrac, predict(Boosting.fit, x.test, type = "prob")[,2])$auc, "\n")
```

```
## Classification Error Rate:
##  Elastic Net: 0.2089249
##  Bagging: 0.1967546
##  Boosting: 0.2068966
##  AUC:
```

```
##  Elastic Net 0.8614035
##  Bagging 0.864333
##  Boosting 0.8676432
```

Bonus: PDPs/ICE

6. (Up to 5 extra points) Present and substantively interpret the "best" model (selected in question 4) using PDPs/ICE curves over the range of: `tolerance` and `age`. Note, interpretation must be more than simple presentation of plots/curves. You must sufficiently describe the changes in *probability* estimates over the range of these two features. You may earn *up to* 5 extra points, where partial credit is possible if the solution is insufficient along some dimension (e.g., technically/code, interpretation, visual presentation, etc.).