Homework4

Lydia Strebe

April 9, 2019

Problem 1: Error Functions for Tree Classification

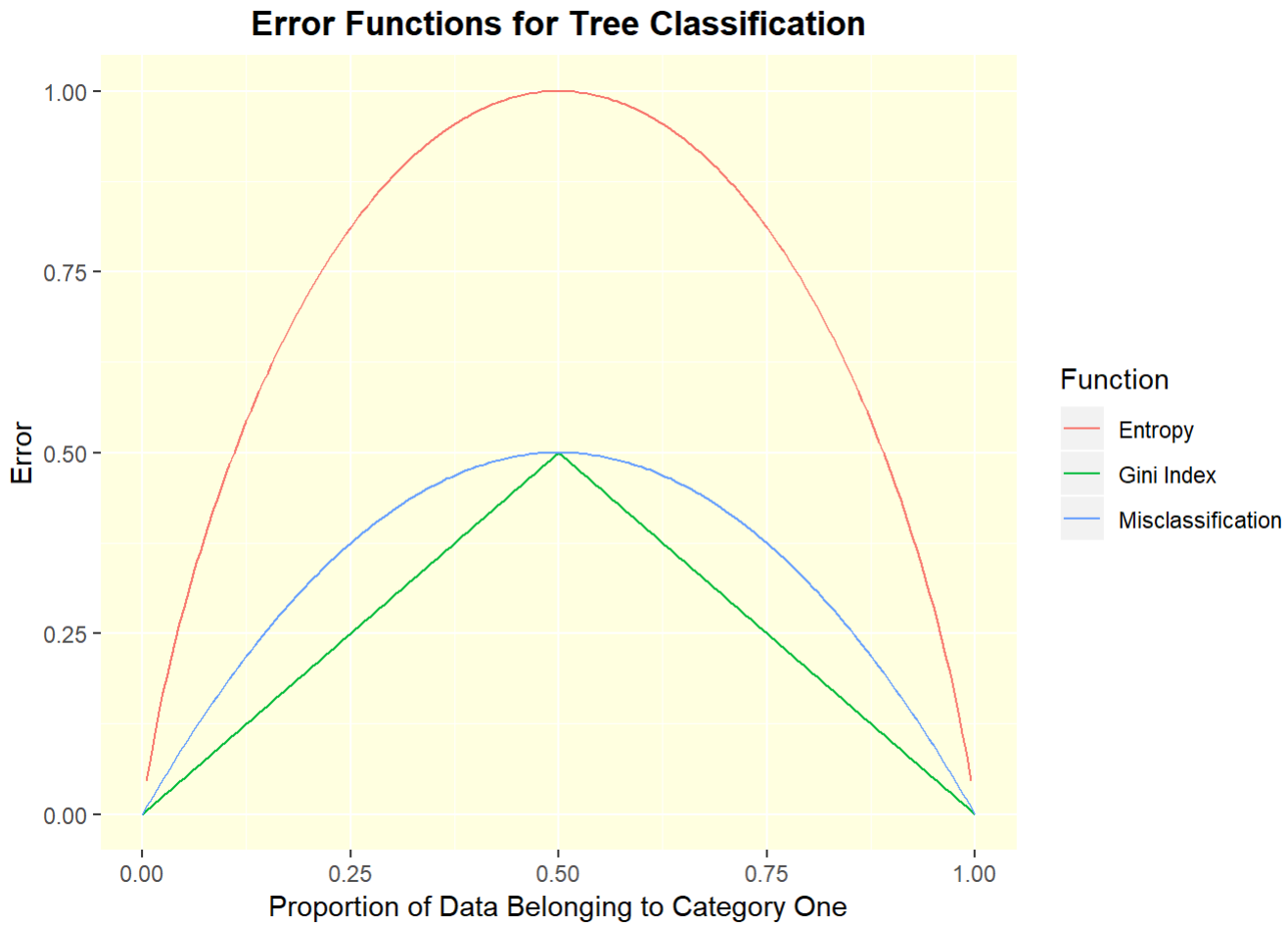
Below is a graph depicting three different error functions in a two-category classification problem:

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
#defining p (proportion of training data points in the mth leaf that belong to category 1)
p=seq(0,1,0.005)

Gini = 2*p*(1-p)

Entropy=(p-1)*log2(1-p)-p*log2(p)

#misclassification
E=0.5-abs(p-0.5)

Error=data.frame(p,E,Gini,Entropy)

ggplot()+geom_line(data=Error,aes(x=p,y=E,col="purple"))->p1
p1+geom_line(data=Error,aes(x=p,y=Gini,col="salmon"))->p2
p2+geom_line(data=Error,aes(x=p,y=Entropy,col="blue"))->p3
p3+theme(panel.background=element_rect(fill="lightyellow"))->p4
p4+theme(plot.title=element_text(hjust=0.5,face="bold"))->p5
p5+labs(title="Error Functions for Tree Classification",y="Error",
        x="Proportion of Data Belonging to Category One")->p6
p6+scale_colour_discrete(name="Function",breaks=c("blue", "purple", "salmon"),
                         labels=c("Entropy", "Gini Index", "Misclassification"))
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```



It is much better to use the Gini index or entropy instead of misclassification error to train trees because of their concave,*curved* shape. Since the algorithm used is greedy (recursive binary splitting), it will only split the tree if the average error of the child nodes is less than the error of the parent node (so that each split lowers the total objective). The average always falls on a straight line between the two child nodes. When using misclassification error, the average error of the child nodes could easily equal the error of the parent node, causing the algorithm to stop prematurely. On the other hand, a line drawn between two points on the Gini index or entropy curve is always under the curve, so the average error will likely not be equal to the parent node.

Problem 2: Predicting Titanic Survivors Using Trees (Kaggle Competition)

Below are the data sets for training and testing a model to predict which passengers on the Titanic survived.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.5.2
```

```
library(MASS)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.5.2
```

```
## Loaded gbm 2.1.5
```

```
Titanic_train=read.csv("C:/Users/lydia/Documents/titanic/train.csv",header=TRUE)
Titanic_test=read.csv("C:/Users/lydia/Documents/titanic/test.csv",header=TRUE)

summary(Titanic_train)
```

```
##   PassengerId       Survived          Pclass
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000
##  Median :446.0   Median :0.0000   Median :3.000
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##                                     Name         Sex           Age
##  Abbing, Mr. Anthony                   :  1   female:314   Min.   : 0.42
##  Abbott, Mr. Rossmore Edward           :  1   male  :577   1st Qu.:20.12
##  Abbott, Mrs. Stanton (Rosa Hunt)      :  1                Median :28.00
##  Abelson, Mr. Samuel                   :  1                Mean   :29.70
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
##  Adahl, Mr. Mauritz Nils Martin        :  1                Max.   :80.00
##  (Other)                               :885                NA's   :177
##      SibSp           Parch             Ticket         Fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33
##                                   (Other) :852
##          Cabin     Embarked
##             :687    :  2
##  B96 B98    :  4   C:168
##  C23 C25 C27:  4   Q: 77
##  G6         :  4   S:644
##  C22 C26    :  3
##  D          :  3
##  (Other)    :186
```

We can remove the column “Passenger Id” from our model since this is, presumably, a number arbitrarily assigned to passengers to organize the data set. We will also remove the “Name” and “Ticket” columns since these inputs are unique and nonnumerical (although the information could be used to engineer new columns). Finally, we will remove the “Cabin” data since it is likewise hard to categorize and most of the information is missing.

We will begin with an AdaBoost model. There is more missing data but the package we use for boosting will take care of it.

```
#AdaBoost

#Take out columns: PassengerId, Name, Ticket, Cabin
Titanic_train_boost=Titanic_train[,-c(1,4,9,11)]
Titanic_test_boost=Titanic_test[,-c(1,3,8,10)]
```

First we tune hyper-parameters using 10-fold cross-validation on depths of 1, 2, and 3 to find the tree with the lowest cross-validation error.

```
set.seed(1)

#tuning hyper-parameters

#depth=1
boost.titanic1=gbm(Survived~.,data=Titanic_train_boost,distribution="adaboost",n.trees=2000,inte
raction.depth=1,cv.folds=10)
cv_err1=boost.titanic1$cv.error
#minimum error
min(cv_err1)
```

```
## [1] 0.7352366
```

```
#best number of trees
which.min(cv_err1)
```

```
## [1] 83
```

```
#depth=2
boost.titanic2=gbm(Survived~.,data=Titanic_train_boost,distribution="adaboost",n.trees=2000,inte
raction.depth=2,cv.folds=10)
cv_err2=boost.titanic2$cv.error
#minimum error
min(cv_err2)
```

```
## [1] 0.7047338
```

```
#best number of trees
which.min(cv_err2)
```

```
## [1] 76
```

```
#depth=3
boost.titanic3=gbm(Survived~.,data=Titanic_train_boost,distribution="adaboost",n.trees=2000,inte
raction.depth=3,cv.folds=10)
cv_err3=boost.titanic3$cv.error
#minimum error
min(cv_err3)
```

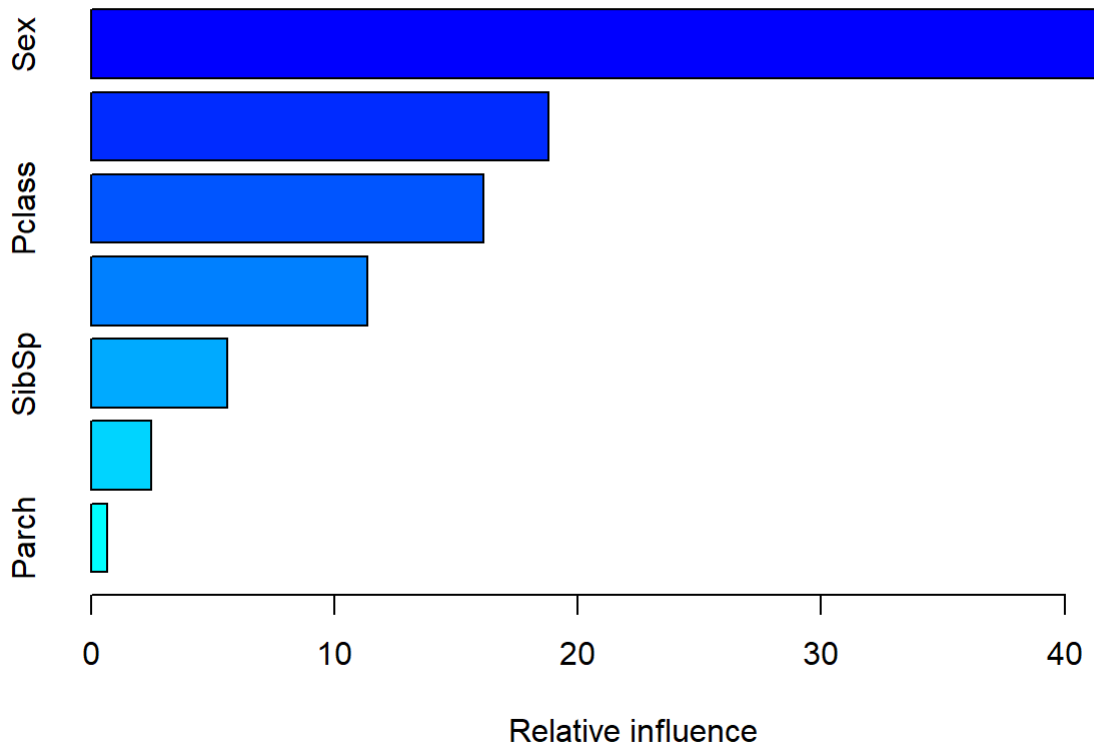
```
## [1] 0.7075063
```

```
#best number of trees
which.min(cv_err3)
```

```
## [1] 53
```

We then use all the training data to build a tree using the best depth/tree combination.

```
#final model
boost.titanic=gbm(Survived~.,data=Titanic_train_boost,distribution="adaboost",n.trees=76,interac
tion.depth=2)
#summary info of final model
summary(boost.titanic)
```




```
##              var    rel.inf
## Sex          Sex 45.0279381
## Age          Age 18.8012406
## Pclass     Pclass 16.1124694
## Fare         Fare 11.3422033
## SibSp       SibSp  5.5949740
## Embarked Embarked  2.4743680
## Parch       Parch  0.6468066
```

Finally, we use our model to make predictions on the test set.

```
#Using test data
#prediction
boost.prediction = predict(boost.titanic,Titanic_test_boost,type="link",n.trees=76,interaction.d
epth=2)
boost.prediction01=ifelse(boost.prediction>0,1,0)

#creating a data frame of the predictions
boost.submission = data.frame(PassengerId = Titanic_test$PassengerId, Survived = boost.predictio
n01)
```

I used the following code to write the output to a csv file so I could upload it to Kaggle:

write.csv(boost.submission, file = “boost_submission.csv”, row.names = FALSE, quote = FALSE)

Next, we will build a Random Forest model. The package for Random Forest does not deal with missing values very well so we will have to clean up the data a bit.

```
#Random Forest

#temporarily removing the Survived column from training data
Titanic_train2=Titanic_train[,-2]

#temorarily combining the test and train data
Titanic_all = rbind(Titanic_train2, Titanic_test)

#summary of data
summary(Titanic_all)
```

```
##   PassengerId       Pclass                                      Name
##  Min.   :   1   Min.   :1.000   Connolly, Miss. Kate            :   2
##  1st Qu.: 328   1st Qu.:2.000   Kelly, Mr. James                :   2
##  Median : 655   Median :3.000   Abbing, Mr. Anthony             :   1
##  Mean   : 655   Mean   :2.295   Abbott, Mr. Rossmore Edward     :   1
##  3rd Qu.: 982   3rd Qu.:3.000   Abbott, Mrs. Stanton (Rosa Hunt):   1
##  Max.   :1309   Max.   :3.000   Abelson, Mr. Samuel             :   1
##                                 (Other)                         :1301
##      Sex           Age            SibSp            Parch
##  female:466   Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  male  :843   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000
##               Median :28.00   Median :0.0000   Median :0.000
##               Mean   :29.88   Mean   :0.4989   Mean   :0.385
##               3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000
##               Max.   :80.00   Max.   :8.0000   Max.   :9.000
##               NA's   :263
##       Ticket          Fare                     Cabin      Embarked
##  CA. 2343:  11   Min.   :  0.000                 :1014    :  2
##  1601    :   8   1st Qu.:  7.896   C23 C25 C27   :   6   C:270
##  CA 2144 :   8   Median : 14.454   B57 B59 B63 B66:   5   Q:123
##  3101295 :   7   Mean   : 33.295   G6            :   5   S:914
##  347077  :   7   3rd Qu.: 31.275   B96 B98       :   4
##  347082  :   7   Max.   :512.329   C22 C26       :   4
##  (Other) :1261   NA's   :1         (Other)       : 271
```

```
str(Titanic_all)
```

```
## 'data.frame':    1309 obs. of  11 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 1307 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 62
9 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 929 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345
133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 187 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

There is a missing fare that we will have to fill in. Since it is only one value (for a male in 3rd class), we will just replace the missing value with 0.

```
#Which passenger is missing fare?
which(is.na(Titanic_all$Fare))
```

```
## [1] 1044
```

```
#passenger missing fare
Titanic_all[1044,]
```

```
##      PassengerId Pclass               Name  Sex  Age SibSp Parch Ticket
## 1044        1044      3 Storey, Mr. Thomas male 60.5     0     0   3701
##      Fare Cabin Embarked
## 1044   NA              S
```

```
#replace missing value with 0
Titanic_all[1044,9] <-0
```

Random Forest can handle missing categorical data, so we don't have to worry about those columns, but there are many ages missing, so we will create age groups instead of using the numerical values.

```
AgeGroup=ifelse(is.na(Titanic_all$Age),"Missing",
               ifelse(Titanic_all$Age<18,"Minor",
                      ifelse(Titanic_all$Age>=18 & Titanic_all$Age<60,"Adult","Senior")))

Titanic_all_age=data.frame(Titanic_all,AgeGroup)
```

As with AdaBoost, we will remove the columns we are not using and then separate the test and train data.

```
#Take out columns: PassengerId, Name, Ticket, Cabin, (Age)
Titanic_all_rf=Titanic_all_age[,-c(1,3,5,8,10)]

#Separate test and train
Titanic_train3=Titanic_all_rf[1:891,]
Titanic_train_rf=data.frame(Titanic_train3,Titanic_train$Survived)
Titanic_test_rf=Titanic_all_rf[892:1309,]
```

We then build the Random Forest model with our training data using a large number of trees and the default mtry (square root of m).

```
set.seed(1)

#Random Forest model
rf.titanic=randomForest(as.factor(Titanic_train.Survived)~.,data=Titanic_train_rf,ntree=2000,imp
ortance=TRUE)

#summary info
importance(rf.titanic)
```

```
##                 0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass    47.16101  61.47351             80.61013         33.23189
## Sex      124.09514 169.21993            171.76763        102.04275
## SibSp     39.97352  11.33856             43.81349         16.39961
## Parch     30.30626  28.89887             41.65307         14.80877
## Fare      45.86204  43.41121             67.29572         66.26678
## Embarked  15.09956  35.52164             38.73803         12.21297
## AgeGroup  33.96474  51.76402             60.88153         20.04685
```

```
varImpPlot(rf.titanic)
```

Finally, we use our model to make predictions on the test set and write the output to a csv file to upload to Kaggle.

```
#prediction
rf.prediction=predict(rf.titanic,Titanic_test_rf)

#data frame of prediction
rf.submission = data.frame(PassengerId = Titanic_test$PassengerId, Survived = rf.prediction)
```

Surprisingly, my Random Forest model did better than my AdaBoost model. My AdaBoost model performed worse than simple models that predict solely by gender. However, my Random Forest did pretty well.

Submission and Description	Public Score
rf_submission.csv a minute ago by Lydia Strebe add submission details	0.77033
boost_submission.csv 3 minutes ago by Lydia Strebe add submission details	0.74641

Results