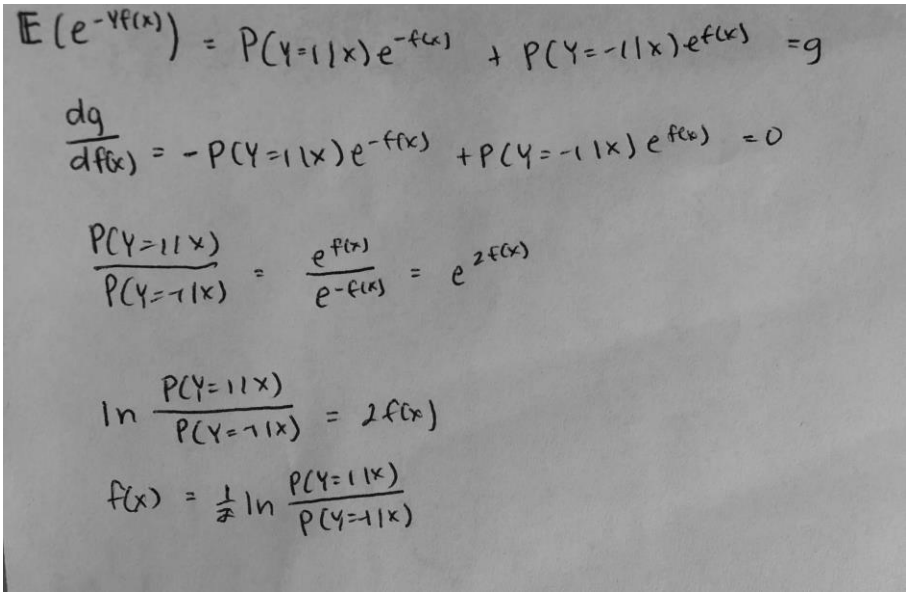
503 HW #4

Diana Liang

3/12/2020

Part 1: Prove Minimizer

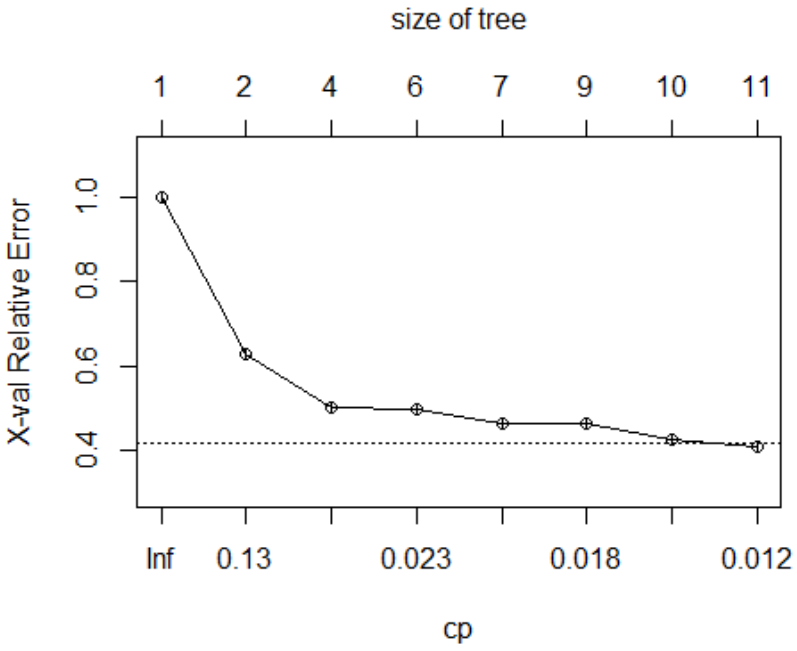


Part 2: Bank Marketing Dataset

```
train <- readr::read_csv('bank_marketing_train.csv')

test <- readr::read_csv('bank_marketing_test.csv')
```

A: Optimal Tree

```
library(rpart)

library(rpart.plot)

library(rattle)

tree_train = rpart(deposit ~ ., train, parms = list(split = "gini"), method =
"class")
plotcp(tree_train)
```



The best cp is 0.016 or lower, so a cp of 0.016 was chosen as the parameter for the optimal tree.

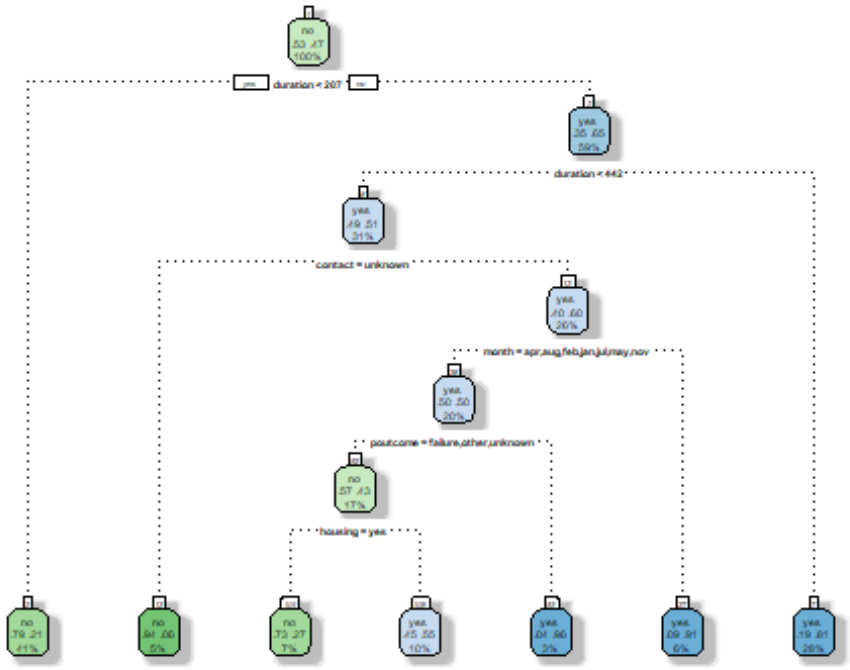
```
tree_opt = rpart(deposit ~ ., train, parms = list(split = "gini"), method =
"class", cp = 0.016)

pred_test = predict(tree_opt, test, type="class")
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.1821439
```

The test error for the optimal tree is shown above.

B: Subtree of Optimal

A larger cp of 0.02 was chosen to create a subtree with fewer terminal nodes.

```
tree_sub = rpart(deposit ~ ., train, parms = list(split = "gini"), method =
"class", cp = 0.02)
fancyRpartPlot(tree_sub)
```



The variables used in the subtree are: duration, contact, month, housing, and poutcome.

C: Random Forest

Original

```
library(randomForest)

train$deposit = factor(train$deposit)
train$job = factor(train$job)
train$marital = factor(train$marital)
train$education = factor(train$education)
train$default = factor(train$default)
train$housing = factor(train$housing)
train$loan = factor(train$loan)
train$contact = factor(train$contact)
train$month = factor(train$month)
train$poutcome = factor(train$poutcome)

test$deposit = factor(test$deposit)
test$job = factor(test$job)
test$marital = factor(test$marital)
test$education = factor(test$education)
test$default = factor(test$default)
```

```
test$housing = factor(test$housing)
test$loan = factor(test$loan)
test$contact = factor(test$contact)
test$month = factor(test$month)
test$poutcome = factor(test$poutcome)

rf_og = randomForest(deposit ~ ., data = train, mtry = ncol(train)-1,
importance = TRUE, ntree = 500)

importance(rf_og)
```

```
##                    no         yes MeanDecreaseAccuracy MeanDecreaseGini
## age         45.1704039  18.2792784           46.4726924       258.606781
## job         26.2099052   9.2720261           24.4209430       215.660342
## marital      2.6274568  11.7143064           11.2051382        41.611931
## education   16.7366910  11.6310102           21.0358626        57.171722
## default     -0.4900215   0.7480235            0.4333794         2.812455
## balance     16.1504930  12.0722331           20.4711591       287.708106
## housing     34.8570842  29.6608384           44.7672996        90.652614
## loan         4.5842227  12.8142792           13.5696856        20.181401
## contact     85.5526270  24.1755311           92.7745242       195.108116
## day         90.3912906  16.5973705           80.9309851       288.734964
## month      164.1412741  60.4780130          173.0546566       595.340990
## duration   227.7371857 276.8231226          330.3957983      1282.032823
## campaign    13.4694373  22.3715059           26.1326672        95.419932
## pdays       26.0278133  24.4160653           35.6761968       108.223071
## previous    18.1868921   1.7743347           18.1752413        47.353941
## poutcome    60.9406022  48.8859140           90.8655201       306.911036
```

The most important variables are: duration, month, poutcome, day, balance, age, and job.

```
pred_test = predict(rf_og, test, type="class")
sum(pred_test != test$deposit)/dim(test)[1]
```

```
## [1] 0.1484025
```

The test error for the original random forest is shown above.

Smaller mtry

```
rf_mtry = randomForest(deposit ~ ., data = train, mtry = 2, importance =
TRUE, ntree = 500)

pred_test = predict(rf_mtry, test, type="class")
sum(pred_test != test$deposit)/dim(test)[1]
```

```
## [1] 0.1487011
```

A smaller mtry had a similar test error to the original since mtry is the number of predictor variables considered for each branching node of the tree which does not have an exact optimal number.

Larger nodesize

```
rf_ndsz = randomForest(deposit ~ ., data = train, mtry = ncol(train)-1,
nodesize = 1000, importance = TRUE, ntree = 500)

pred_test = predict(rf_ndsz, test, type="class")
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.2096148
```

A larger nodesize had a higher test error, since nodesize controls how big each tree can get which may lead to overfitting the training data.

Smaller ntrees

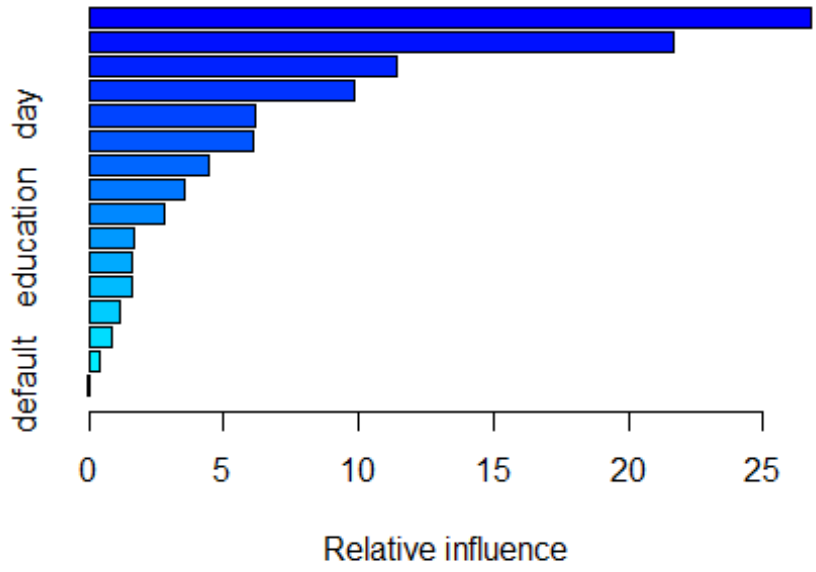
```
rf_nt = randomForest(deposit ~ ., data = train, mtry = ncol(train)-1,
importance = TRUE, ntree = 20)

pred_test = predict(rf_nt, test, type="class")
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.1552702
```

Smaller ntrees had a higher test error, since ntrees is the number of trees that are produced so having too few trees does not capture the full training set as well.

D: Boosting

Original

```
library(gbm)

train$deposit = ifelse(train$deposit=="yes",1,0)
test$deposit = ifelse(test$deposit=="yes",1,0)

ada_og = gbm(deposit~., data = train, distribution = "adaboost", n.trees =
5000, interaction.depth = 3)

summary(ada_og)
```




```
##                  var     rel.inf
## duration    duration 26.79114112
## month          month 21.67229265
## job              job 11.41757233
## balance      balance  9.84796634
## day              day  6.17611735
## age              age  6.10898865
## poutcome    poutcome  4.47410407
## pdays          pdays  3.50564675
## contact      contact  2.79447123
## education education  1.66731715
## housing      housing  1.58492111
## campaign    campaign  1.56509980
## marital      marital  1.12569107
## previous    previous  0.83220868
## loan             loan  0.39744168
## default      default  0.03902003
```

The most important variables are: duration, month, job, and balance.

```
pred_test = predict(ada_og, newdata = test, n.trees = 5000, type =
"response")
pred_test = ifelse(pred_test>0.5,1,0)
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.1501941
```

The test error is shown above.

Greater interaction.depth

```
ada_int = gbm(deposit~., data = train, distribution = "adaboost", n.trees =
5000, interaction.depth = 5)

pred_test = predict(ada_int, newdata = test, n.trees = 5000, type =
"response")
pred_test = ifelse(pred_test>0.5,1,0)
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.1489997
```

Greater interaction.depth has similar test error, since interaction.depth controls how big each tree is allowed to get. At even greater interaction.depth, the trees would be allowed to be much bigger and might overfit to the training data and result in greater test error.

Larger shrinkage

```
ada_sh = gbm(deposit~., data = train, distribution = "adaboost", n.trees =
5000, interaction.depth = 3, shrinkage = 0.5)

pred_test = predict(ada_sh, newdata = test, n.trees = 5000, type =
"response")
pred_test = ifelse(pred_test>0.5,1,0)
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.171693
```

Larger shrinkage yielded greater test error since shrinkage controls how heavily the classifiers are weighted. Usually the smallest test error would result from the shrinkage parameter being 0.1 or 0.01.

Smaller n.trees

```
ada_nt = gbm(deposit~., data = train, distribution = "adaboost", n.trees =
20, interaction.depth = 3)

pred_test = predict(ada_nt, newdata = test, n.trees = 20, type = "response")
pred_test = ifelse(pred_test>0.5,1,0)
sum(pred_test != test$deposit)/dim(test)[1]

## [1] 0.1797552
```

Smaller n.trees yielded greater test error since n.trees is the number of trees produced and too few trees would not properly sample all the data fully.