1. $E_{Y|X}[e^{-Yf(x)}] = P(Y=1|x)e^{-f(x)} + P(Y=-1|x)e^{f(x)}$, $\quad Y = \left\{ \begin{matrix} -1 \\ 1 \end{matrix} \right.$

Define $a = e^{f(x)}$ $\quad a \geqslant 0$, then

$E_{Y|X}[e^{-Yf(x)}] = P(Y=1|x)\frac{1}{a} + P(Y=-1|x)a = F(a)$

$$\frac{\partial F(a)}{\partial a} = P(Y=1|x)(-\frac{1}{a^2}) + P(Y=-1|x)$$

set $\frac{\partial F(a)}{\partial a} = 0 \Rightarrow P(Y=-1|x) = \frac{1}{a^2}P(Y=1|x)$

$$a = \left(\frac{P(Y=1|x)}{P(Y=-1|x)}\right)^{\frac{1}{2}}$$

So, $e^{f(x)} = \left(\frac{P(Y=1|x)}{P(Y=-1|x)}\right)^{\frac{1}{2}}$

$$f(x) = \frac{1}{2}\ln\left(\frac{P(Y=1|x)}{P(Y=-1|x)}\right)$$

Since $\frac{\partial^2 F(a)}{\partial^2 a} = \frac{2}{a^3} > 0$ by $a \geqslant 0$, $f(x) = \frac{1}{2}\ln\left(\frac{P(Y=1|x)}{P(Y=-1|x}\right)$ is the minimal of $E_{Y|X}(e^{-Yf(x)})$.

Therefore, $f^*(x) = \underset{f(x)}{\operatorname{argmin}}\, E_{Y|X}(e^{-Yf(x)}) = \frac{1}{2}\log\left(\frac{Pr(Y=1|x)}{Pr(Y=-1|x)}\right)$

HW4_503

Ningyuan Wang

3/12/2020

Question 2

We split data into training and testing sets and check the values of each variable. There is no werid patterns of variable based on checking.

Based on the context of the problem, besides of overall testing error, we think the banking is also interested in if the product (bank term deposit) would be subscribed. Therefore, we tried to minimize the overall testing error and testing error among all "yes"s, because they don't want to lose their potential clients and want to perform better service for potential clients.

```r
# load the data and split into training and testing sets
set.seed(503)
train = read.csv("bank_marketing_train.csv", header = T)
test = read.csv("bank_marketing_test.csv", header = T)

# EDA
table(train$deposit)
```

```
##
##   no  yes
## 4128 3685
```

```r
summary(train)
```

```
##       age                 job           marital          education
##  Min.   :18.00   management :1779   divorced: 911   primary  :1039
##  1st Qu.:32.00   blue-collar:1346   married :4452   secondary:3882
##  Median :39.00   technician :1275   single  :2450   tertiary :2535
##  Mean   :41.34   admin.     : 937                   unknown  : 357
##  3rd Qu.:49.00   services   : 648
##  Max.   :95.00   retired    : 560
##                  (Other)    :1268
##  default       balance      housing     loan          contact
##  no :7694   Min.   :-6847   no :4138   no :6776   cellular :5647
##  yes: 119   1st Qu.:  122   yes:3675   yes:1037   telephone: 555
##             Median :  539                         unknown  :1611
##             Mean   : 1500
##             3rd Qu.: 1693
##             Max.   :81204
##
##       day            month         duration         campaign
##  Min.   : 1.00   may    :1957   Min.   :   4.0   Min.   : 1.000
##  1st Qu.: 8.00   aug    :1074   1st Qu.: 138.0   1st Qu.: 1.000
##  Median :15.00   jul    :1065   Median : 253.0   Median : 2.000
##  Mean   :15.55   jun    : 830   Mean   : 368.4   Mean   : 2.489
##  3rd Qu.:21.00   nov    : 692   3rd Qu.: 486.0   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 631   Max.   :3881.0   Max.   :41.000
##                   (Other):1564
##      pdays           previous          poutcome    deposit
##  Min.   : -1.00   Min.   : 0.0000   failure: 853   no :4128
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 363   yes:3685
##  Median : -1.00   Median : 0.0000   success: 760
##  Mean   : 50.74   Mean   : 0.8147   unknown:5837
##  3rd Qu.: 14.00   3rd Qu.: 1.0000
##  Max.   :842.00   Max.   :55.0000
##
```

###a. Tree Model

In the optimal tree model with Gini splitting, the overall testing error was 18.12%; the error rate among all "no" clients was 23.78%; the error rate among all "yes" clients was 11.97%.

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
# use gini
tree1 = rpart(deposit ~., train, parms = list(split = "gini"), method = "class")
```

```
# test error
test.pred = predict(tree1, test, type = "class")
sum(test.pred!=test$deposit) / dim(test)[1] # test error
```

```
## [1] 0.1812481
```

```
# misclassification among "no"s
test.no = test[test$deposit=="no", ]
sum(test.pred[1605:3349]!="no") / dim(test.no)[1] #0.2378223
```

```
## [1] 0.2378223
```

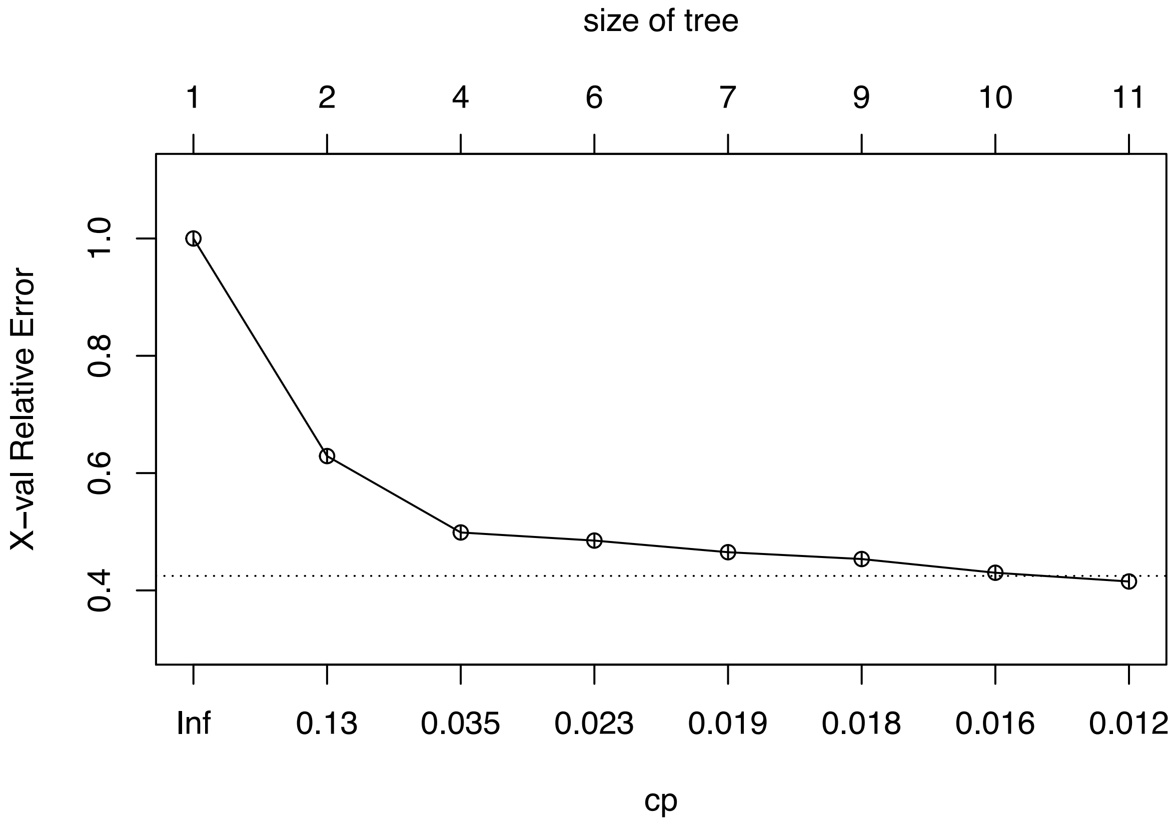
```
# misclassfication among "yes"s
test.yes = test[test$deposit=="yes", ]
sum(test.pred[1:1604]!="yes")/dim(test.yes)[1] #0.1197007
```

```
## [1] 0.1197007
```

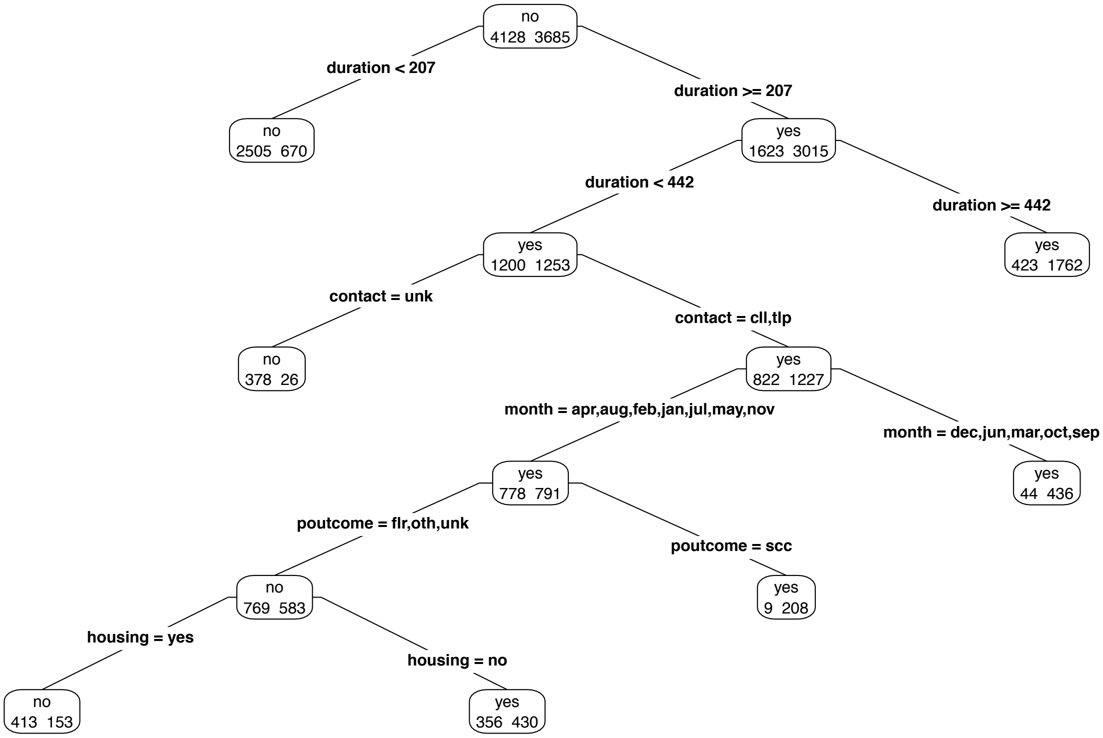
###b.

We tried several models to prune the terminal nodes under 8, which cp should be larger than 0.019. The plot of subtree of the optimal tree shown as follow. In this tree model, the useful variables were duration, contact, month and housing.

```
plotcp(tree1)
```



```
tree2 = rpart(deposit ~., train, parms = list(split = "gini"), method = "class", cp = 0.019)
prp(tree2, type = 4, extra = 1, clip.right.labs = F)
```



###c. Random Forest

We tried several random forest models and controlled the effect of mtry, nodesize and ntree in different levels. mtry refers to the number of variables randomly sampled as candidates at each split, and the default value was square root of predictors. Nodesize refers to the minimum size of terminal nodes, the larger node size caused less computing time. Ntree refers to the number of trees to grow, which being preferred a large number to ensure that every input row gets predicted at least a few times.

We tried models back and forth. For example, controlled the effect of mtry and nodesize, increasing the ntree from 500 to 1000 will increase the testing error; controlled the effect of mtry and ntree, increasing the nodesize from 1 to 2 will decrease the testing error.

Finally, the optimal model gave us overall testing error at 14.30%, the error among all "no" was 17.77%, and the error among all "yes" was 10.54%. In general, the random forest models improved the performance than using single tree. The useful variables in the optimal model were duration, month, contact and balance.

```
library(randomForest)
```

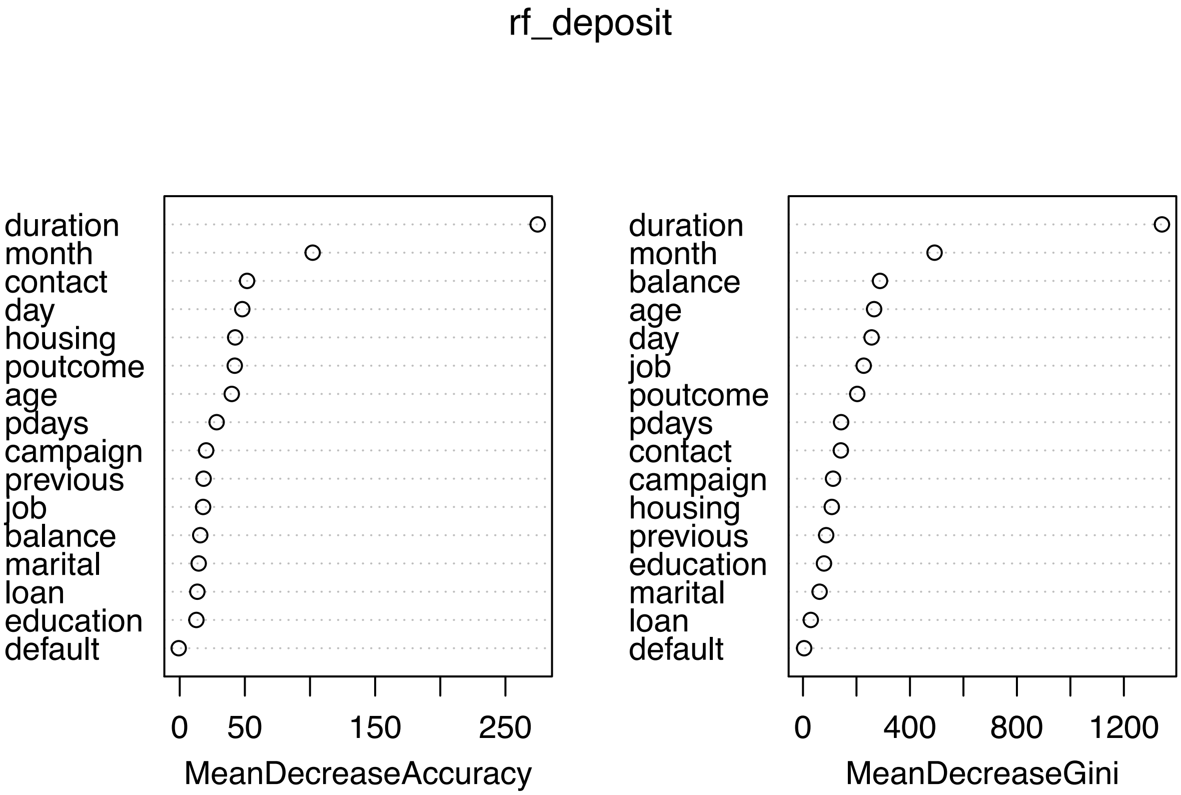
```
## randomForest 4.6-14
```

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

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

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
set.seed(503)
rf_deposit = randomForest(deposit~., data = train, importance = TRUE, ntree = 500) # use 4 classifier
# importance(rf_deposit)
varImpPlot(rf_deposit)
```



```
# test performance
rf_test_pred = predict(rf_deposit, newdata = test)
table(rf_test_pred, test$deposit)
```

```
##
## rf_test_pred   no  yes
##          no  1435  169
##          yes  310 1435
```

```
rf_test_err = mean(rf_test_pred!=test$deposit)
rf_test_err
```

```
## [1] 0.1430278
```

```
# misclassification among "no"s
test.no = test[test$deposit=="no", ]
sum(rf_test_pred[1605:3349]!="no") / dim(test.no)[1]
```

```
## [1] 0.1776504
```

```
# misclassfication among "yes"s
test.yes = test[test$deposit=="yes", ]
sum(rf_test_pred[1:1604]!="yes")/dim(test.yes)[1]
```

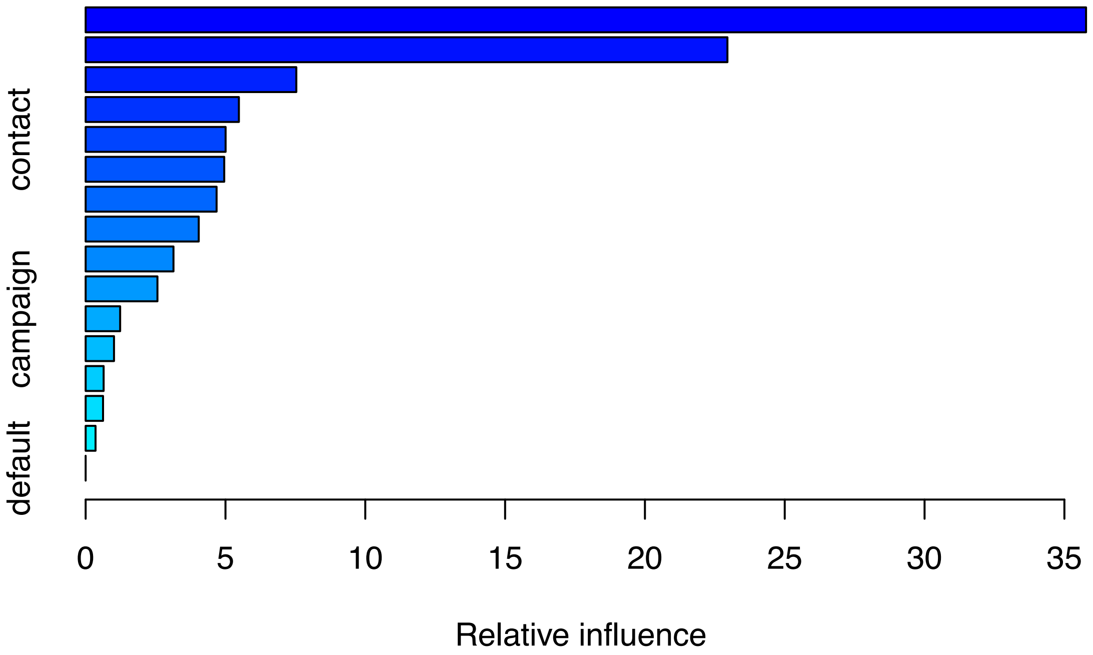
```
## [1] 0.1053616
```

###d. Boosting We applied AdaBoost algorithm and controlled depth, shrinkage and ntrees in different levels. We found that the decrease of shrinkage will train a better model and therefore better performance in the testing error. However, large ntrees and depth will cause overfitting issue, which may increase the testing error. The optimal model by Adaboost gave us the testing error at 13.97%; the error among all "no" s was 13.97% and the error among all "yes"s was 14.61%. The useful variables were duration, month, poutcome, job and contact.

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(503)
train$deposit = ifelse(train$deposit=="yes",1,0)
test$deposit = ifelse(test$deposit=="yes",1,0)
ada_deposit = gbm(deposit~., data = train, distribution = "adaboost", n.trees = 500, interaction.depth =
summary(ada_deposit,)
```



```
##                 var    rel.inf
## duration   duration 35.7751340
## month         month 22.9503653
## poutcome   poutcome  7.5330844
## job             job  5.4798191
## contact     contact  5.0046743
## balance     balance  4.9557910
## day             day  4.6823099
## age             age  4.0438682
## pdays         pdays  3.1412346
## housing     housing  2.5672260
## campaign   campaign  1.2332577
## education education  1.0142317
## marital     marital  0.6434524
## previous   previous  0.6216028
## loan           loan  0.3539486
## default     default  0.0000000
```

```
ada_pred_response = predict(ada_deposit, newdata = test,type = "response", n.trees = 500)


# test performance
ada_pred = ifelse(ada_pred_response>0.5,1,0)
table(ada_pred,test$deposit)
```

```
##
## ada_pred    0    1
##        0 1490  213
##        1  255 1391
```

```
ada_test_err = mean(ada_pred!=test$deposit)
ada_test_err # 0.1397432
```

```
## [1] 0.1397432
```

```
# misclassification among "no"s
mean(ada_pred[1605:3349]!=0)  # 0.1397432
```

```
## [1] 0.1461318
```

```
# misclassfication among "yes"s
mean(ada_pred[1:1604]!=1)  #0.1461318
```

```
## [1] 0.132793
```