EnsableAssn.R

SeanOMalley1

Wed May 6 10:18:11 2015

```
# Ensamble Models in R
# Sean O'Malley

bank <- read.csv("/Users/SeanOMalley1/Desktop/Week\ 7\ ADM/bank-full.csv")

library(lattice)
library(ggplot2)
library(gplots)
```

```
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(mlbench)
library(plyr)
library(datasets)
library(graphics)
library(grDevices)
library(methods)
library(stats)
library(utils)
library(caret)
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.1.3
```

```
library(gmodels)
library(doSNOW)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: snow
```

```
library(adabag)
library(rpart.plot)

# We are determining wheather someone is subscribing to a fixed term deposit to our b
ank. The classifcation model we are creating is going to determine the question, "Wha
t type of people subscribe to fixed term deposits?"
# Dependant Variable is "subscribed" variable
#The advantage of such a deposit is that the bank doesn't have to worry about the ind
ividual taking any money out for a fixed amount of time, thus providing more financia
l options and guaranteeing the money will be available to the bank for a longer perio
d of time.

# EDA
str(bank) # classification random forest ensamble model where the dependant variable
is y..category
```

```
## 'data.frame':    45211 obs. of  17 variables:
##  $ age       : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job       : Factor w/ 12 levels "admin.","blue-collar",..: 5 10 3 2 12 5 5 3 6
10 ...
##  $ marital   : Factor w/ 3 levels "divorced","married",..: 2 3 2 2 3 2 3 1 2 3 ...
##  $ education : Factor w/ 4 levels "primary","secondary",..: 3 2 2 4 4 3 3 3 1 2 ..
.
##  $ default   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ balance   : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing   : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ loan      : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ contact   : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3 3 3 3 3 3 3 .
..
##  $ day       : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month     : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ duration  : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays     : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ outcome   : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ subscribed: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
names(bank)
```

```
##  [1] "age"        "job"        "marital"    "education"  "default"   
##  [6] "balance"    "housing"    "loan"       "contact"    "day"       
## [11] "month"      "duration"   "campaign"   "pdays"      "previous"  
## [16] "outcome"    "subscribed"
```

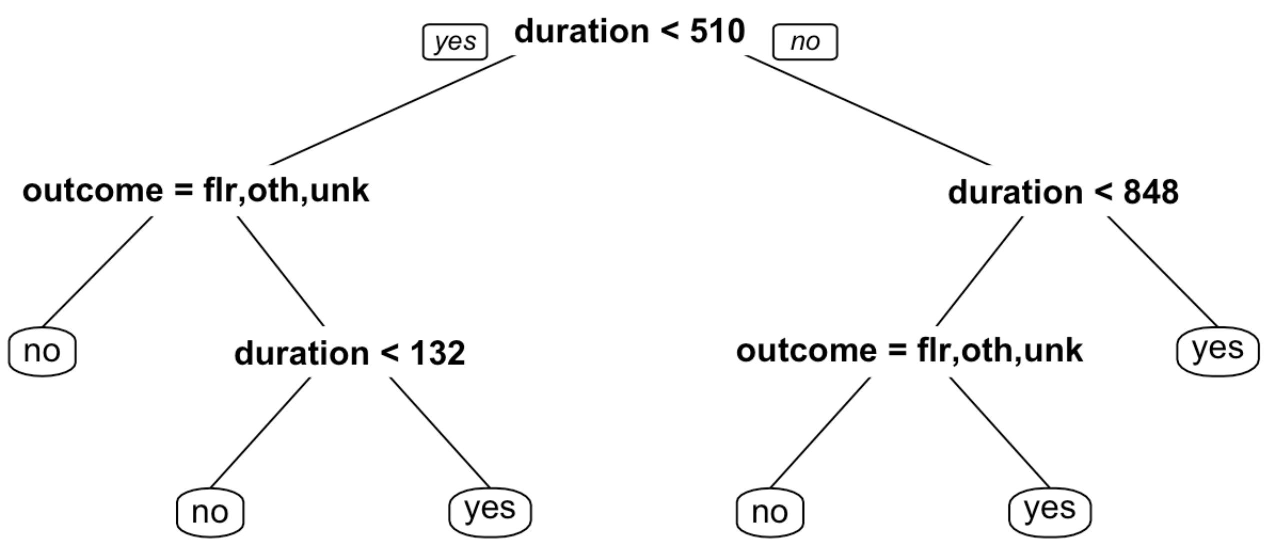
```
summary(bank) # many unknown variables
```

```
##       age                 job            marital          education    
##  Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851  
##  1st Qu.:33.00   management :9458   married :27214   secondary:23202  
##  Median :39.00   technician :7597   single  :12790   tertiary :13301  
##  Mean   :40.94   admin.     :5171                    unknown  : 1857  
##  3rd Qu.:48.00   services   :4154                                     
##  Max.   :95.00   retired    :2264                                     
##                  (Other)    :6835                                     
##  default        balance       housing      loan            contact     
##  no :44396   Min.   : -8019   no :20081   no :37967   cellular :29285  
##  yes:  815   1st Qu.:    72   yes:25130   yes: 7244   telephone: 2906  
##              Median :   448                           unknown  :13020  
##              Mean   :  1362                                            
##              3rd Qu.:  1428                                            
##              Max.   :102127                                            
##                                                                        
##       day            month          duration         campaign     
##  Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000  
##  1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000  
##  Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000  
##  Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764  
##  3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000  
##  Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000  
##                  (Other): 6060                                     
##      pdays          previous            outcome      subscribed 
##  Min.   : -1.0   Min.   :  0.0000   failure: 4901   no :39922  
##  1st Qu.: -1.0   1st Qu.:  0.0000   other  : 1840   yes: 5289  
##  Median : -1.0   Median :  0.0000   success: 1511              
##  Mean   : 40.2   Mean   :  0.5803   unknown:36959              
##  3rd Qu.: -1.0   3rd Qu.:  0.0000                              
##  Max.   :871.0   Max.   :275.0000                              
## 
```

```
##################################################

# Model 1: Dec Tree # All Variables
            # Train and Test/Holdout Examples
set.seed(99)
bank_rand1 <- bank[order(runif(45211)),] #45,211 total observations
bank_rand1 <- sample(1:45211, 31648) #want 31,648 obs for training and 13,563 obs in
test. that is about a 70/30
bank_train1 <- bank[ bank_rand1,]
bank_test1  <- bank[-bank_rand1,]

# Decision Tree: # (rpart)
bank_tree1 <- rpart(subscribed~.,data=bank_train1)
rpart.plot(bank_tree1)
```




```
print(bank_tree1)
```

```
## n= 31648
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 31648 3710 no (0.88277300 0.11722700)
##    2) duration< 510.5 28071 2134 no (0.92397848 0.07602152)
##      4) outcome=failure,other,unknown 27171 1581 no (0.94181296 0.05818704) *
##      5) outcome=success 900  347 yes (0.38555556 0.61444444)
##       10) duration< 132.5 178   43 no (0.75842697 0.24157303) *
##       11) duration>=132.5 722  212 yes (0.29362881 0.70637119) *
##    3) duration>=510.5 3577 1576 no (0.55940732 0.44059268)
##      6) duration< 847.5 2401  880 no (0.63348605 0.36651395)
##       12) outcome=failure,other,unknown 2281  781 no (0.65760631 0.34239369) *
##       13) outcome=success 120   21 yes (0.17500000 0.82500000) *
##      7) duration>=847.5 1176  480 yes (0.40816327 0.59183673) *
```

```
summary(bank_tree1) # shows duration and outcome as only important factors, could mea
n parameter adjustment needed
```

```
## Call:
## rpart(formula = subscribed ~ ., data = bank_train1)
##   n= 31648
##
##           CP nsplit rel error    xerror       xstd
## 1 0.03791554      0 1.0000000 1.0000000 0.01542544
## 2 0.02479784      3 0.8862534 0.8900270 0.01465843
## 3 0.02102426      4 0.8614555 0.8638814 0.01446621
## 4 0.01000000      5 0.8404313 0.8455526 0.01432902
##
## Variable importance
## duration  outcome
##       62       37
##
## Node number 1: 31648 observations,    complexity param=0.03791554
##   predicted class=no   expected loss=0.117227  P(node) =1
##     class counts: 27938  3710
##    probabilities: 0.883 0.117
##   left son=2 (28071 obs) right son=3 (3577 obs)
##   Primary splits:
##       duration < 510.5 to the left,     improve=843.3836, (0 missing)
##       outcome  splits as  LLRL,         improve=622.3676, (0 missing)
##       month    splits as  LLRLLLLRLLRR, improve=335.0443, (0 missing)
##       pdays    < 8.5   to the left,     improve=190.5850, (0 missing)
##       previous < 0.5   to the left,     improve=187.1104, (0 missing)
```

```
##
## Node number 2: 28071 observations,    complexity param=0.03791554
##   predicted class=no   expected loss=0.07602152  P(node) =0.8869755
##     class counts: 25937  2134
##    probabilities: 0.924 0.076
##   left son=4 (27171 obs) right son=5 (900 obs)
##   Primary splits:
##       outcome  splits as  LLRL,         improve=539.1031, (0 missing)
##       month    splits as  LLRLLLLRLLRR, improve=314.3254, (0 missing)
##       pdays    < 16    to the left,     improve=173.0977, (0 missing)
##       previous < 0.5   to the left,     improve=169.8065, (0 missing)
##       duration < 206.5 to the left,     improve=146.7630, (0 missing)
##
## Node number 3: 3577 observations,    complexity param=0.03791554
##   predicted class=no   expected loss=0.4405927  P(node) =0.1130245
##     class counts:  2001  1576
##    probabilities: 0.559 0.441
##   left son=6 (2401 obs) right son=7 (1176 obs)
##   Primary splits:
##       duration < 847.5 to the left,     improve=80.15318, (0 missing)
##       outcome  splits as  LLRL,         improve=51.35249, (0 missing)
##       contact  splits as  RRL,          improve=46.33695, (0 missing)
##       month    splits as  LRRLLLLRLLRR, improve=35.23122, (0 missing)
##       pdays    < 8.5   to the left,     improve=22.24908, (0 missing)
##   Surrogate splits:
##       campaign < 22.5  to the left,  agree=0.672, adj=0.003, (0 split)
##       previous < 17.5  to the left,  agree=0.672, adj=0.003, (0 split)
##       age      < 87.5  to the left,  agree=0.672, adj=0.001, (0 split)
##       balance  < -1207 to the right, agree=0.672, adj=0.001, (0 split)
##       pdays    < 392.5 to the left,  agree=0.672, adj=0.001, (0 split)
##
## Node number 4: 27171 observations
##   predicted class=no   expected loss=0.05818704  P(node) =0.8585377
##     class counts: 25590  1581
##    probabilities: 0.942 0.058
##
## Node number 5: 900 observations,    complexity param=0.02479784
##   predicted class=yes  expected loss=0.3855556  P(node) =0.02843782
##     class counts:   347   553
##    probabilities: 0.386 0.614
##   left son=10 (178 obs) right son=11 (722 obs)
##   Primary splits:
##       duration < 132.5 to the left,     improve=61.698340, (0 missing)
##       housing  splits as  RL,           improve=11.996250, (0 missing)
##       month    splits as  RRRRRRRRLLRR, improve= 8.053067, (0 missing)
##       pdays    < 51.5  to the left,     improve= 7.023146, (0 missing)
##       campaign < 3.5   to the right,    improve= 5.919221, (0 missing)
```

```
##   Surrogate splits:
##       contact splits as  RRL, agree=0.810, adj=0.039, (0 split)
##       default splits as  RL,  agree=0.803, adj=0.006, (0 split)
##
## Node number 6: 2401 observations,    complexity param=0.02102426
##   predicted class=no   expected loss=0.366514  P(node) =0.07586577
##     class counts:  1521   880
##    probabilities: 0.633 0.367
##   left son=12 (2281 obs) right son=13 (120 obs)
##   Primary splits:
##       outcome  splits as  LLRL,         improve=53.10438, (0 missing)
##       contact  splits as  RRL,          improve=35.94764, (0 missing)
##       month    splits as  LRRLLLLRLLRR, improve=32.56742, (0 missing)
##       pdays    < 8.5   to the left,     improve=27.50869, (0 missing)
##       previous < 0.5   to the left,     improve=27.39978, (0 missing)
##
## Node number 7: 1176 observations
##   predicted class=yes  expected loss=0.4081633  P(node) =0.03715875
##     class counts:   480   696
##    probabilities: 0.408 0.592
##
## Node number 10: 178 observations
##   predicted class=no   expected loss=0.241573  P(node) =0.005624368
##     class counts:   135    43
##    probabilities: 0.758 0.242
##
## Node number 11: 722 observations
##   predicted class=yes  expected loss=0.2936288  P(node) =0.02281345
##     class counts:   212   510
##    probabilities: 0.294 0.706
##
## Node number 12: 2281 observations
##   predicted class=no   expected loss=0.3423937  P(node) =0.07207406
##     class counts:  1500   781
##    probabilities: 0.658 0.342
##
## Node number 13: 120 observations
##   predicted class=yes  expected loss=0.175  P(node) =0.003791709
##     class counts:    21    99
##    probabilities: 0.175 0.825
```

```
#confusion matrix for rpart
bank_tree1_actual <- bank_test1$subscribed #created to test the "test" data/
bank_tree1_pred <- predict(bank_tree1, bank_test1, type="class")
bank_tree1_results <- confusionMatrix(bank_tree1_pred, bank_tree1_actual) #the model
vs the actual holdout data.
print(bank_tree1_results)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11668  1033
##        yes   316   546
##
##                Accuracy : 0.9005
##                  95% CI : (0.8954, 0.9055)
##     No Information Rate : 0.8836
##     P-Value [Acc > NIR] : 1.705e-10
##
##                   Kappa : 0.3978
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9736
##             Specificity : 0.3458
##          Pos Pred Value : 0.9187
##          Neg Pred Value : 0.6334
##              Prevalence : 0.8836
##          Detection Rate : 0.8603
##    Detection Prevalence : 0.9364
##       Balanced Accuracy : 0.6597
##
##        'Positive' Class : no
##
```

```
# We have 90% accuracy at predicting whether someone will subscribe to fixed term dep
osits
# Greedy algorithm is only using 2/17 variables available to make a prediction, need
further research

#################################################

# Model 2 : Random Forest
set.seed(99)
bank_rf1 <- randomForest(subscribed ~.,data=bank_train1, ntree=25,na.action = na.omit
, importance=TRUE)
# can put ~.because he got rid of id #, see line 6
print(bank_rf1) #shows OOB of model and confusion matrix
```

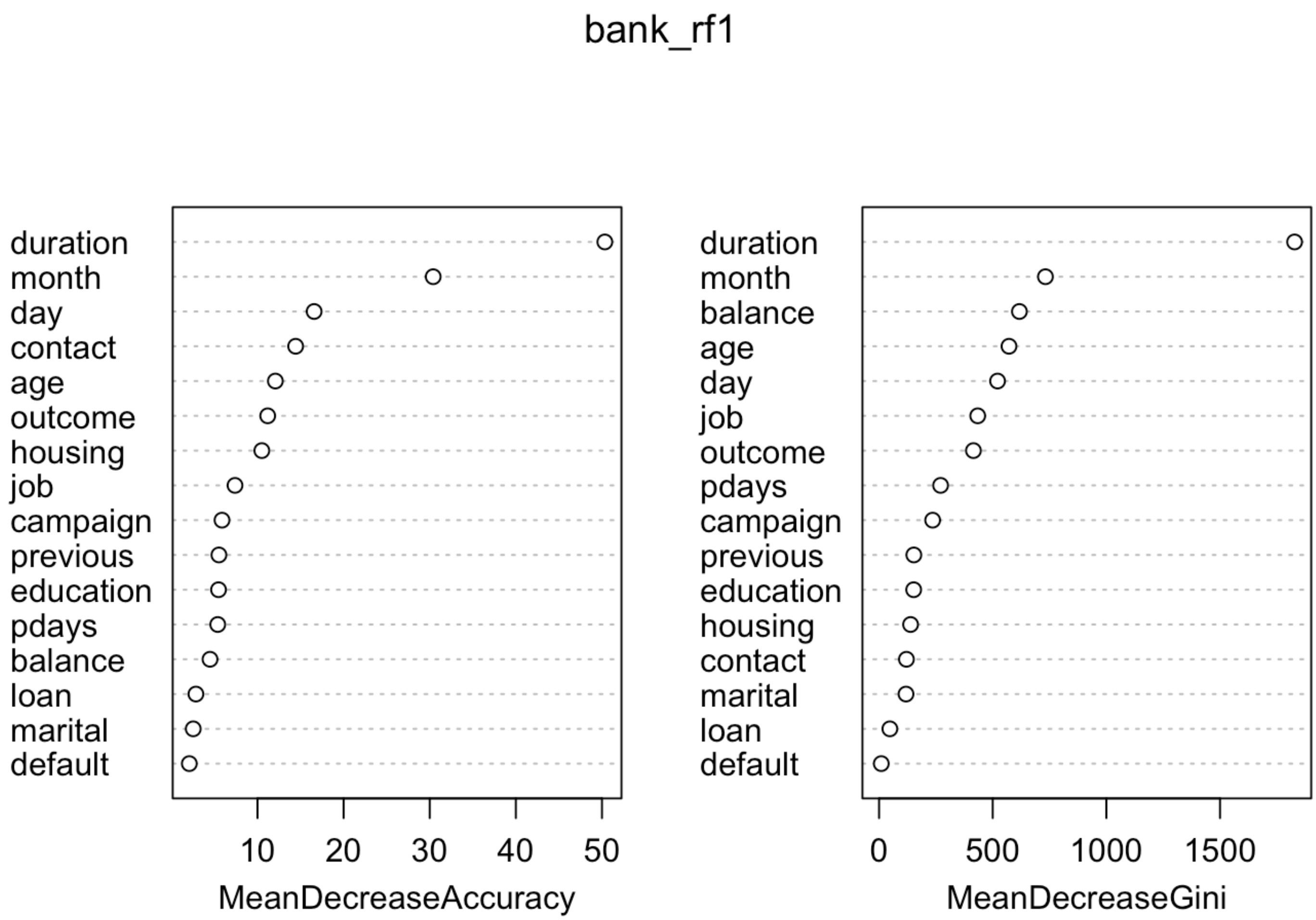
```
##
## Call:
##  randomForest(formula = subscribed ~ ., data = bank_train1, ntree = 25,      impor
tance = TRUE, na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 25
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.91%
## Confusion matrix:
##        no  yes class.error
## no  26710 1227  0.04392025
## yes  1910 1800  0.51482480
```

```
importance(bank_rf1) #shows the importance of each variable
```

```
##                   no       yes MeanDecreaseAccuracy MeanDecreaseGini
## age        10.063115  4.7133183            12.075863       571.611929
## job         7.134809  0.2691294             7.389245       433.964363
## marital     1.245914  2.9150187             2.539107       118.530072
## education   5.753903  0.7716123             5.475338       152.246258
## default     1.071415  1.9177811             2.085381         9.729627
## balance     3.892177  2.3841880             4.489242       617.421018
## housing    10.222948  6.3030120            10.504654       138.627167
## loan        1.037465  3.2801608             2.851221        47.517952
## contact    13.708937  1.5313865            14.431085       120.340680
## day        16.939956  2.5380681            16.578163       521.310875
## month      27.221931  7.7196258            30.402505       731.948972
## duration   27.755439 60.0978345            50.333508      1828.074949
## campaign    5.165356  2.6466831             5.879911       235.754048
## pdays       4.807660  5.1166029             5.384774       270.696341
## previous    5.266209  3.4538595             5.521918       152.959713
## outcome     7.859392  3.4875573            11.185021       415.124405
```

```
# No. of variables tried at each split: 4 # at each branch it picks 4/17 variables, t
hen pics best one for tree.
# then after the trees are ensembled it takes the best of 500 trees that are picky
# OOB estimate of error rate: tells us that the average error rate of all trees was 9
.31% aka 90.69% accuracy
# confusion matrix inaccuracy shows this is all run off of training data, yet to do t
est data
# unbalanced dataset with majority being true negatives

varImpPlot(bank_rf1) #plots the importance of each variable
```




```
# unlike the decision tree, default, loan and balance affect the accuracy of the dv t
he most
# Gini measures entropy, accuracy is most important, further left = more important
# tells us most important features to tell manager this way, doesn't give split point
s bc theyre synthetic

# Running test through test data
bank_rf1_actual <- bank_test1$subscribed
bank_rf1_pred <-predict(bank_rf1, bank_test1,type="response")
bank_rf1_results <- confusionMatrix(bank_rf1_pred, bank_rf1_actual)
print(bank_rf1_results)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11509   795
##        yes   475   784
##
##                Accuracy : 0.9064
##                  95% CI : (0.9013, 0.9112)
##     No Information Rate : 0.8836
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.501
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9604
##             Specificity : 0.4965
##          Pos Pred Value : 0.9354
##          Neg Pred Value : 0.6227
##              Prevalence : 0.8836
##          Detection Rate : 0.8486
##    Detection Prevalence : 0.9072
##       Balanced Accuracy : 0.7284
##
##        'Positive' Class : no
##
```

```
# accuracy improves marginally, but kappa and sensitivity show the greatest improveme
nts
# making this model better than the tree model
CrossTable(bank_rf1_pred, bank_rf1_actual)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  13563
##
##
##              | bank_rf1_actual
## bank_rf1_pred |        no |       yes | Row Total |
## --------------|-----------|-----------|-----------|
##            no |     11509 |       795 |     12304 |
##              |    37.374 |   283.654 |           |
##              |     0.935 |     0.065 |     0.907 |
##              |     0.960 |     0.503 |           |
##              |     0.849 |     0.059 |           |
## --------------|-----------|-----------|-----------|
##           yes |       475 |       784 |      1259 |
##              |   365.250 |  2772.105 |           |
##              |     0.377 |     0.623 |     0.093 |
##              |     0.040 |     0.497 |           |
##              |     0.035 |     0.058 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |     11984 |      1579 |     13563 |
##              |     0.884 |     0.116 |           |
## --------------|-----------|-----------|-----------|
##
##
```

```
# true postives and negatives show great increase in comparison to previous model

# K Fold Cross Validation of Model 2
registerDoSNOW(makeCluster(2, type = "SOCK")) #sets to use the 2 cores in my laptop
ctrl <- trainControl(method = "repeatedcv",
                     number = 5, repeats = 5) #sets crossvalidation run kfold against
itself
# creates kfold of 10 # this syntax essentially just sets tuning 10^10
# each fold and each cross validation is parallelizable

set.seed(99)
bank_rf1_cv <- train(subscribed ~.,data=bank_train1, method = "rf",
                       metric = "Kappa", trControl = ctrl) #method is randomforest.
#Adds in the ctrl as control of the line above
# takes 500 trees and randomly tries multiple tries at trees

print(bank_rf1_cv)
```

```
## Random Forest
##
## 31648 samples
##    16 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
##
## Summary of sample sizes: 25318, 25319, 25318, 25319, 25318, 25319, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.8924039  0.1682657  0.001006159  0.01345966
##   22    0.9048786  0.4805970  0.002650910  0.01814799
##   42    0.9042593  0.4819962  0.003013329  0.01994296
##
## Kappa was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 42.
```

```
# mtry is the approximate square of how many splits
# showed us random splits that gave us the highest kappas, which is what we asked for
# grid search: goes through a range of # and find what works best
#use all your other RF diagnostics and plots here
     # 5 splits (22mtry) is the optimal amount of kappa at 0.483, with 0.905 accuracy

# Auto-tune random forest
bank_rf1_grid <- expand.grid(.mtry = c(4,9,25))
#mtry is how many variables it randomly tries at each node (sqrt of mtry is # of spli
ts)

set.seed(99)
bank_ptm <- proc.time() #starts to time
bank_rf1_cv2 <- train(subscribed ~.,data=bank_train1, method = "rf", ntree=500,
                        metric = "Kappa", trControl = ctrl, tuneGrid = bank_rf1_grid)
#also control mtry
print(bank_rf1_cv2)
```

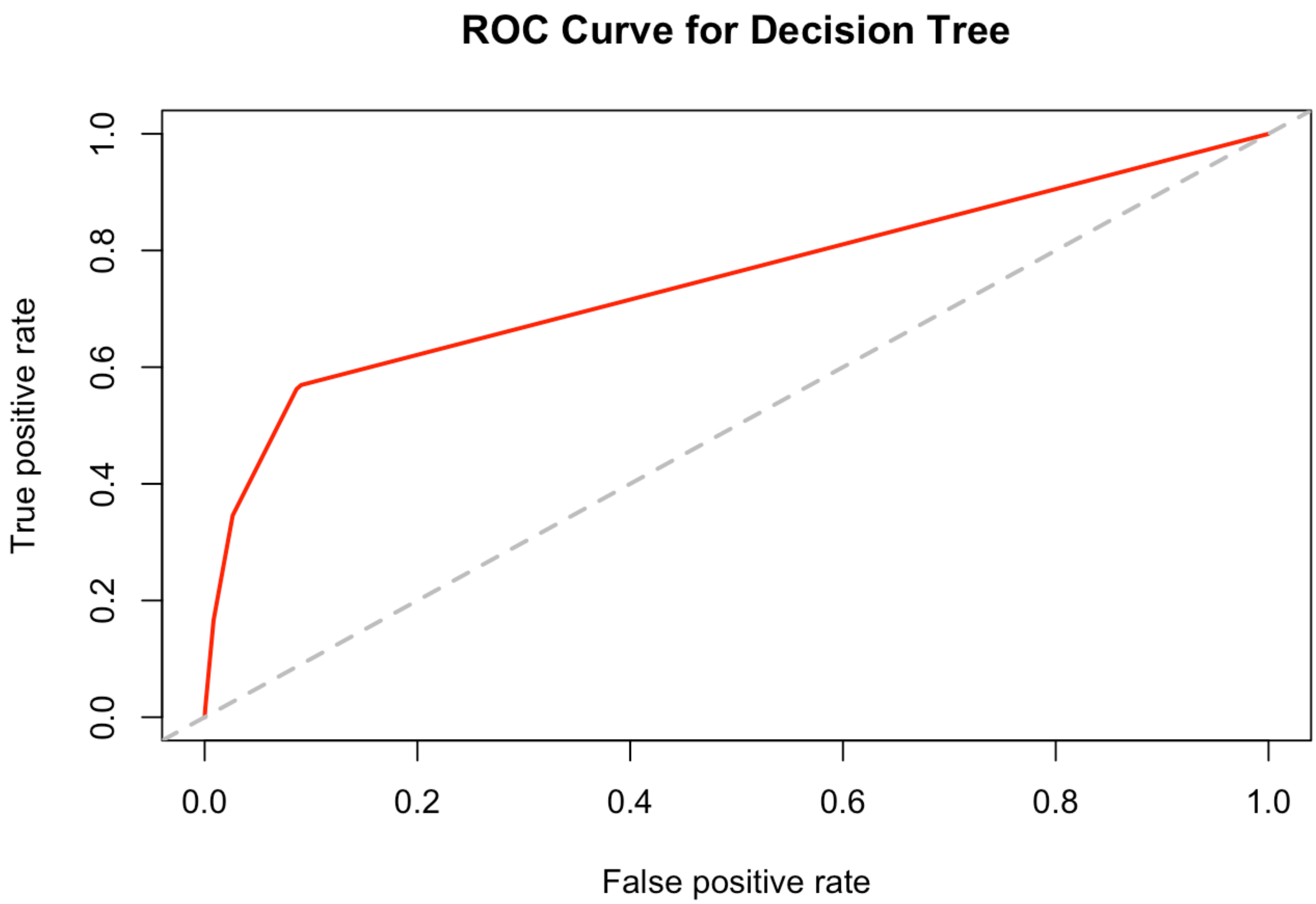
```
## Random Forest
##
## 31648 samples
##    16 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
##
## Summary of sample sizes: 25318, 25319, 25318, 25319, 25318, 25319, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    4    0.9025847  0.3592532  0.001548192  0.01712334
##    9    0.9057633  0.4679619  0.002682394  0.01870807
##   25    0.9048533  0.4822413  0.002975948  0.02019670
##
## Kappa was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 25.
```

```
# after adding ctrl and tune
# mtry is how many variables it randomly tries at each node (sqrt of mtry is # of spl
its)
# if you want the highest kappa, use the corresponding mtry to find optimal number of
splits
# could go back and do a grid search to find best kmeans

summary(bank_rf1_cv2)
```

```
##                 Length Class      Mode
## call                5  -none-     call
## type                1  -none-     character
## predicted       31648  factor     numeric
## err.rate         1500  -none-     numeric
## confusion           6  -none-     numeric
## votes           63296  matrix     numeric
## oob.times       31648  -none-     numeric
## classes             2  -none-     character
## importance         42  -none-     numeric
## importanceSD        0  -none-     NULL
## localImportance     0  -none-     NULL
## proximity           0  -none-     NULL
## ntree               1  -none-     numeric
## mtry                1  -none-     numeric
## forest             14  -none-     list
## y               31648  factor     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames             42  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           2  -none-     character
```

```
# ROC Curve for Decision Tree
bank_tree1_pred_prob1 <-predict(bank_tree1, type="prob", bank_test1)# same as above f
or predict, but add "prob".
bank_pred2 <- prediction(bank_tree1_pred_prob1[,2],bank_test1$subscribed)
bank_perf2 <- performance(bank_pred2,"tpr","fpr") #true pos and false pos
plot(bank_perf2 ,main="ROC Curve for Decision Tree",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#area under the curve
bank_tree1_auc <- performance(bank_pred2, measure = "auc") #run an area under the cur
ve
str(bank_tree1_auc) #see different values in the auc object
```

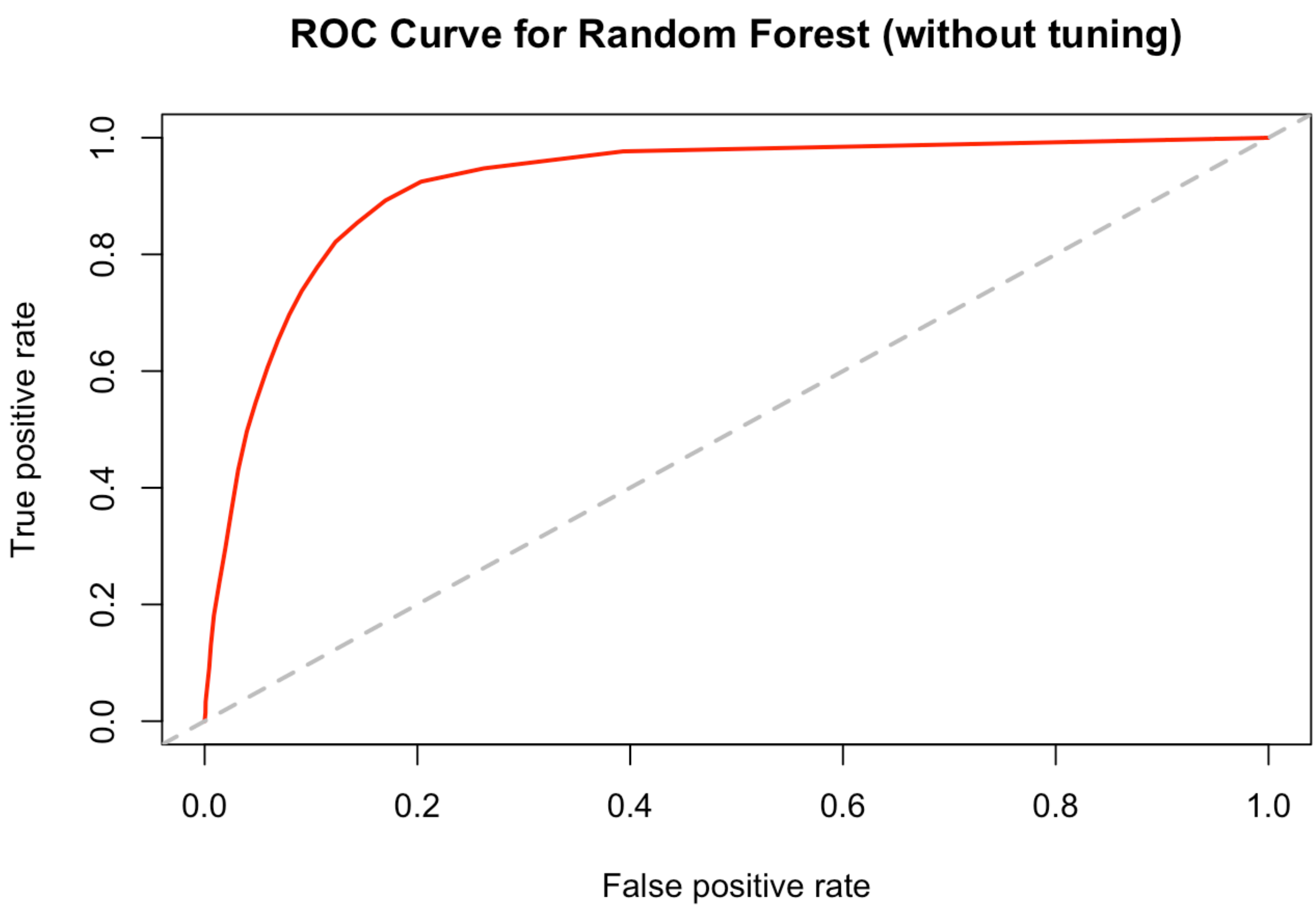
```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.749
##   ..@ alpha.values: list()
```

```
as.numeric(bank_tree1_auc@y.values) #shows the AUC percentage
```

```
## [1] 0.7485122
```

```
    # 74% above

# ROC curve for RF1
bank_rf1_pred_prob1 <-predict(bank_rf1, type="prob", bank_test1)# same as above for p
redict, but add "prob".
bank_pred1 <- prediction(bank_rf1_pred_prob1[,2],bank_test1$subscribed)
bank_perf1 <- performance(bank_pred1,"tpr","fpr") #true pos and false pos
plot(bank_perf1 ,main="ROC Curve for Random Forest (without tuning)",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#area under the curve
bank_rf1_auc <- performance(bank_pred1, measure = "auc") #run an area under the curve
str(bank_rf1_auc) #see different values in the auc object
```

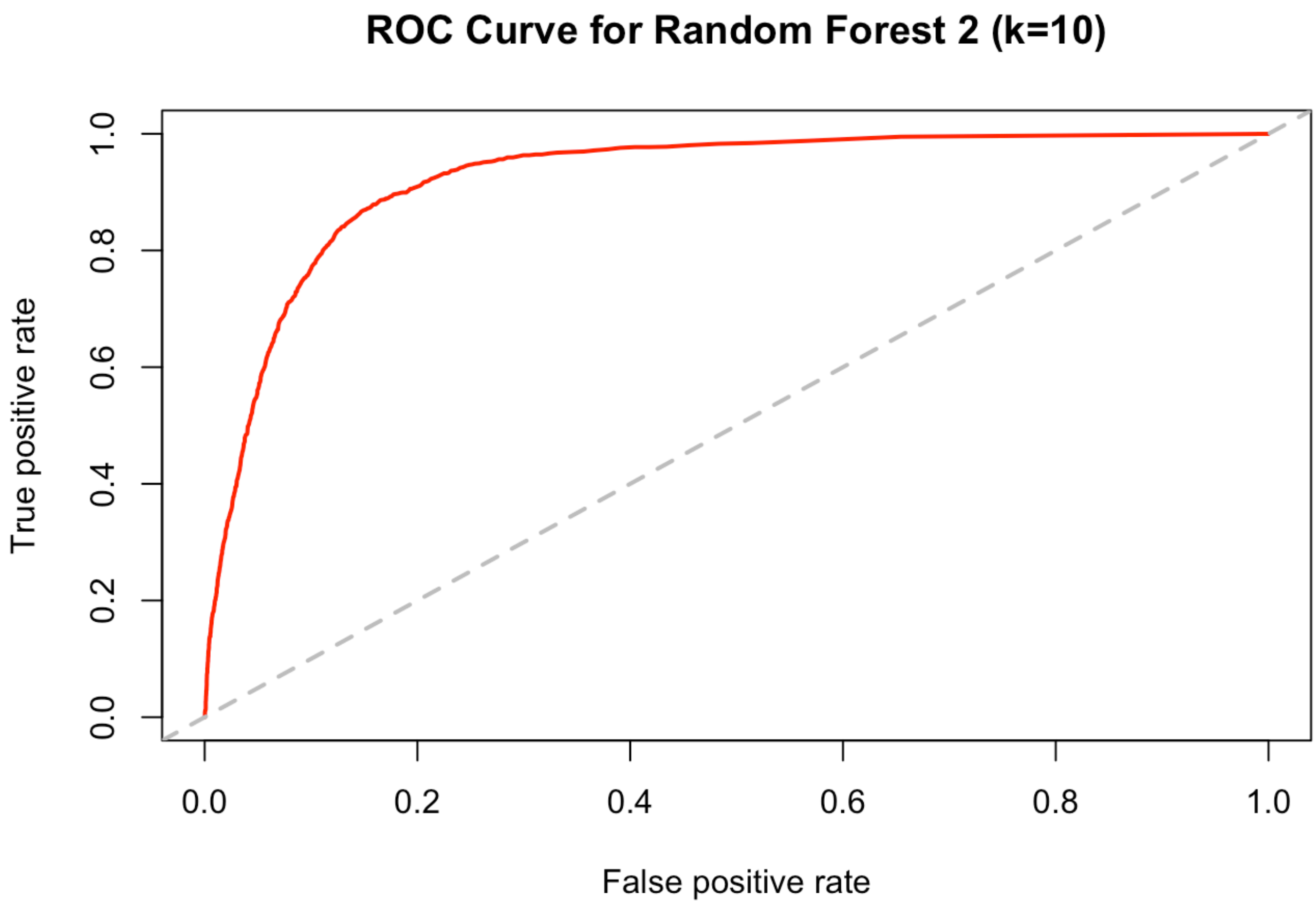
```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.92
##   ..@ alpha.values: list()
```

```
as.numeric(bank_rf1_auc@y.values) #shows the AUC percentage
```

```
## [1] 0.9202552
```

```
    # 93% above

# ROC curve for RF2
bank_rf2_pred_prob1 <-predict(bank_rf1_cv, type="prob", bank_test1)# same as above fo
r predict, but add "prob".
bank_pred3 <- prediction(bank_rf2_pred_prob1[,2],bank_test1$subscribed)
bank_perf3 <- performance(bank_pred3,"tpr","fpr") #true pos and false pos
plot(bank_perf3 ,main="ROC Curve for Random Forest 2 (k=10)",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#area under the curve
bank_rf2_auc <- performance(bank_pred3, measure = "auc") #run an area under the curve
str(bank_rf2_auc) #see different values in the auc object
```

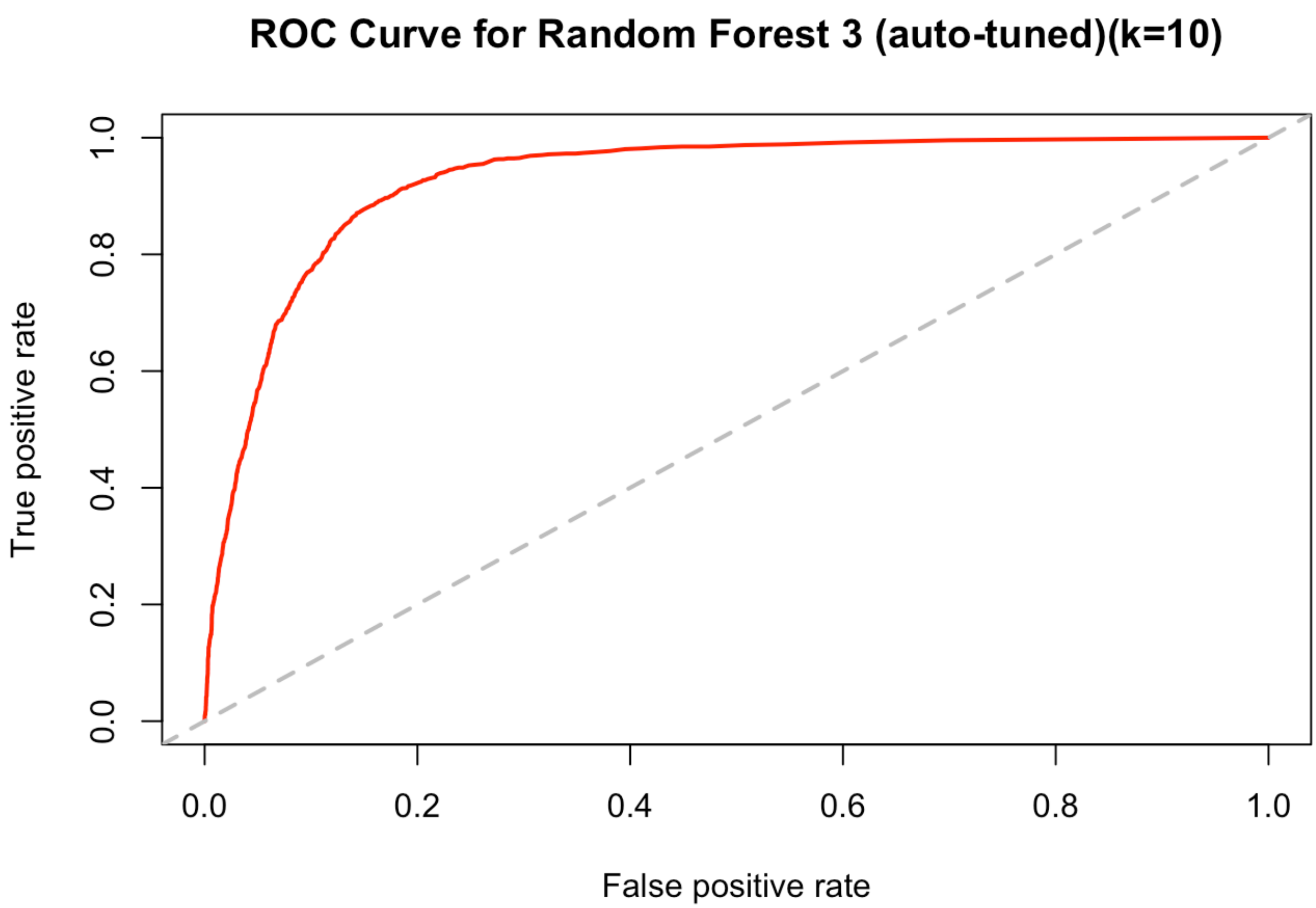
```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.924
##   ..@ alpha.values: list()
```

```
as.numeric(bank_rf2_auc@y.values) #shows the AUC percentage
```

```
## [1] 0.9239141
```

```
     # 92.7% above (computationally expensive)

# ROC curve for RF3
bank_rf3_pred_prob1 <-predict(bank_rf1_cv2, type="prob", bank_test1)# same as above f
or predict, but add "prob".
bank_pred4 <- prediction(bank_rf3_pred_prob1[,2],bank_test1$subscribed)
bank_perf4 <- performance(bank_pred4,"tpr","fpr") #true pos and false pos
plot(bank_perf4 ,main="ROC Curve for Random Forest 3 (auto-tuned)(k=10)",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#area under the curve
bank_rf3_auc <- performance(bank_pred4, measure = "auc") #run an area under the curve
str(bank_rf3_auc) #see different values in the auc object
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.927
##   ..@ alpha.values: list()
```

```
as.numeric(bank_rf3_auc@y.values) #shows the AUC percentage
```

```
## [1] 0.9271857
```

```
     # 92.6% above (computationally expensive)
```