library(randomForest)

rf\_model <- randomForest(success~., data=train)

rf\_pred <- predict(rf\_model, test, type='response')

cm <- confusionMatrix(rf\_pred, test$success)

cm

library(kernlab)

svm\_rbf <- ksvm(success~., data=train ,kernel = "vanilladot")

pred\_rbf <- predict(svm\_rbf, select(test, -success))

agreement\_rbf <- pred\_rbf == test$success

> table(agreement\_rbf)

agreement\_rbf

FALSE  TRUE

  193   334

> prop.table(table(agreement\_rbf))

agreement\_rbf

    FALSE      TRUE

0.3662239 0.6337761

cm <- confusionMatrix(pred\_rbf, test$success)

cm

DT\_model <- C5.0(train, train\_label$train.success)

DT\_model

> summary(DT\_model)

Call:

C5.0.default(x = train, y = train\_label$train.success)

C5.0 [Release 2.07 GPL Edition]   Thu May  9 04:38:33 2024

-------------------------------

Class specified by attribute `outcome``

Read 2107 cases (18 attributes) from undefined.data

Decision tree:

rating <= 0.5135135: 0 (896/211)

rating > 0.5135135:

:...duration > 0.3904639: 0 (26/3)

    duration <= 0.3904639:

    :...is\_stellar = 1: 0 (23/5)

        is\_stellar = 0:

        :...teamSize <= 0.1351351: 0 (415/150)

            teamSize > 0.1351351:

            :...rating > 0.8108108: 1 (166/50)

                rating <= 0.8108108:

                :...hasReddit = 0: 0 (110/41)

                    hasReddit = 1: 1 (471/219)

Evaluation on training data (2107 cases):

      Decision Tree

    ----------------

    Size      Errors

       7  679(32.2%)   <<

     (a)   (b)    <-classified as

    ----  ----

    1060   269    (a): class 0

     410   368    (b): class 1

  Attribute usage:

  100.00% rating

   57.48% duration

   56.24% is\_stellar

   55.15% teamSize

   27.57% hasReddit

Time: 0.0 secs

# predict the model

DT\_pred <- predict(DT\_model, test)

# Evaluation

library(gmodels)

CrossTable(DT\_pred, test\_label$test.success,

           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

           dnn = c('predicted', 'actual' ))

library(caret)

cm <- confusionMatrix(DT\_pred, test\_label$test.success)

cm

Appendix: Evaluation

sample\_size <- floor(0.8 \* nrow(ico\_testdata))

set.seed(123)

train\_ind <- sample(nrow(ico\_testdata), sample\_size)

train <- ico\_testdata[train\_ind, ]

test <- ico\_testdata[-train\_ind, ]

seed <- 7

metric <- "Accuracy"

set.seed(seed)

control <- trainControl(method="cv", number=5,search='grid')

tunegrid <- expand.grid(.mtry=c(1:10))

modellist <- list()

for (ntree in c(300,400,500,600,700,800)) {

  set.seed(seed)

  fit <- train(success~., data=train, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control, ntree=ntree)

  key <- toString(ntree)

  modellist[[key]] <- fit

}

# compare results

results <- resamples(modellist)

summary(results)

best <- modellist['500']

best

$`500`

Random Forest

2100 samples

  17 predictor

   2 classes: '0', '1'

No pre-processing

Resampling: Cross-Validated (5 fold)

Summary of sample sizes: 1679, 1680, 1680, 1680, 1681

Resampling results across tuning parameters:

  mtry  Accuracy   Kappa

   1    0.6285706  0.00682849

   2    0.6552249  0.17960663

   3    0.6547555  0.19666484

   4    0.6452317  0.17783602

   5    0.6461886  0.18492617

   6    0.6433360  0.17834971

   7    0.6419040  0.17729483

   8    0.6457135  0.18567411

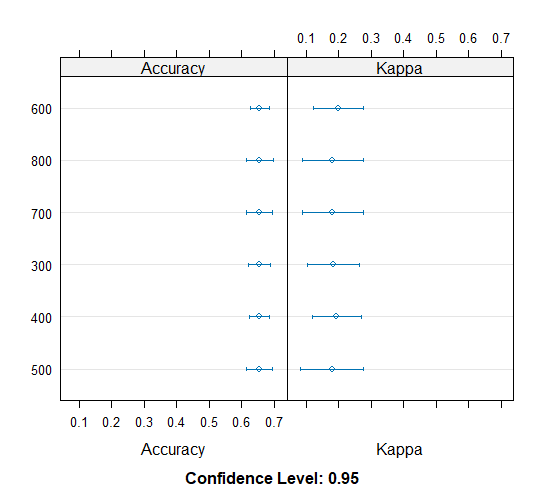
   9    0.6404754  0.17557894

  10    0.6347532  0.16602391

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2.

dotplot(results)



# --- FINAL EVALUATION

> set.seed(300)

> library(randomForest)

> control <- trainControl(search='grid')

> tunegrid <- expand.grid(mtry=c(1:10))

> rf\_model <- randomForest(success~., data=train, ntree=500, tuneGrid=tunegrid, trControl=control)

> rf\_model

> rf\_pred <- predict(rf\_model, test, type='response')

> rf\_pred\_object <- predict(rf\_model, test, type='prob')

> cm <- confusionMatrix(rf\_pred, test$success)

> cm

pred\_object <- prediction(rf\_pred\_object[,2], test$success)

RF\_Eval <- performance(pred\_object, measure = "tpr", x.measure = "fpr")

plot(RF\_Eval, main = "ROC curve for success ico projects", col = "blue", lwd = 2)

abline(a = 0, b = 1, lwd = 2, lty = 2)

auc <- performance(pred\_object, measure = "auc")

# get the AUC value

auc\_RF <- auc@y.values[[1]]

auc\_RF

# 0.6822514