Hello, my name is influencer

Ceccotti Alex, Fiorini Stefano, Govi Davide, Sessi Michela

Dataset: 5500 righe, 23 variabili (https://www.kaggle.com/c/predict-who-is-more-influential-in-a-social-network)

Target binario Choice: quale tra i due utenti A e B è più influente (1=A, 0=B)

```r
d <- read.csv("~/data_science_lab/train.csv")
status=df_status(d, print_results = F)
pander(status[,-c(6,7)]%>%arrange(type,-q_zeros))
```

variable	q_zeros	p_zeros	q_na	p_na	type	unique
Choice	2698	49.05	0	0	integer	2
A_network_feature_1	212	3.85	0	0	integer	345
B_network_feature_1	194	3.53	0	0	integer	350
A_listed_count	60	1.09	0	0	integer	523
B_listed_count	46	0.84	0	0	integer	528
A_following_count	35	0.64	0	0	integer	673
B_following_count	32	0.58	0	0	integer	668
A_follower_count	0	0	0	0	integer	759
B_follower_count	0	0	0	0	integer	760
A_network_feature_2	346	6.29	0	0	numeric	691
B_network_feature_2	341	6.2	0	0	numeric	708
A_network_feature_3	260	4.73	0	0	numeric	733
B_network_feature_3	247	4.49	0	0	numeric	743
A_mentions_received	0	0	0	0	numeric	719
A_retweets_received	0	0	0	0	numeric	582
A_mentions_sent	0	0	0	0	numeric	500
A_retweets_sent	0	0	0	0	numeric	247
A_posts	0	0	0	0	numeric	553
B_mentions_received	0	0	0	0	numeric	732
B_retweets_received	0	0	0	0	numeric	590
B_mentions_sent	0	0	0	0	numeric	518
B_retweets_sent	0	0	0	0	numeric	257
B_posts	0	0	0	0	numeric	581

```r
prop.table(table(d$Choice))
```

```
##
##         0         1
## 0.4905455 0.5094545
```

```r
for (i in (2:12)){
  d[,(i+22)] <- d[,i]/d[,(11+i)]
  names(d)[i+22] <- paste("rapp", substring(names(d)[i], 2), sep="")
}
status=df_status(d, print_results = F)
pander(head(status[,c(1,4,5)]%>%arrange(-q_na)))
```

variable	q_na	p_na
rapp_network_feature_2	17	0.31

variable	q_na	p_na
rapp_network_feature_3	7	0.13
rapp_network_feature_1	4	0.07
Choice	0	0
A_follower_count	0	0
A_following_count	0	0

```
pander(head(status[,c(1,6,7)]%>%arrange(-q_inf)))
```

variable	q_inf	p_inf
rapp_network_feature_2	324	5.89
rapp_network_feature_3	240	4.36
rapp_network_feature_1	190	3.45
rapp_listed_count	46	0.84
rapp_following_count	32	0.58
Choice	0	0

```
for (i in (24:ncol(d))){
  d[is.na(d[,i]),i] <- 1
  d[d[,i]==Inf,i] <- d[d[,i]==Inf,(i-22)]
}
status=df_status(d, print_results = F)
pander(head(status[,c(1,4,5)]%>%arrange(-q_na)))
```

variable	q_na	p_na
Choice	0	0
A_follower_count	0	0
A_following_count	0	0
A_listed_count	0	0
A_mentions_received	0	0
A_retweets_received	0	0

```
pander(head(status[,c(1,6,7)]%>%arrange(-q_inf)))
```

variable	q_inf	p_inf
Choice	0	0
A_follower_count	0	0
A_following_count	0	0
A_listed_count	0	0
A_mentions_received	0	0
A_retweets_received	0	0

```
train<-d

train$A_foll_ratio <- train$A_following_count/train$A_follower_count
train$A_ment_ratio <- train$A_mentions_sent/train$A_mentions_received
train$A_retw_ratio <- train$A_retweets_sent/train$A_retweets_received
```

```r
train$B_foll_ratio <- train$B_following_count/train$B_follower_count
train$B_ment_ratio <- train$B_mentions_sent/train$B_mentions_received
train$B_retw_ratio <- train$B_retweets_sent/train$B_retweets_received
```

```r
train$A_zeros <- 0
train$B_zeros <- 0
for (i in (2:12)){
  train$A_zeros[train[,i]==0] <- train$A_zeros[train[,i]==0] + 1
}
for (i in (13:23)){
  train$B_zeros[train[,i]==0] <- train$B_zeros[train[,i]==0] + 1
}

train$has_zeros <- FALSE
train$has_zeros[(train$A_zeros+train$B_zeros)>0]<-TRUE
```

```r
train$Choice=ifelse(train$Choice==1,"A","B")
dim(train)
```

```
## [1] 5500   43
```

```r
pander(summary(train))
```

Table 6: Table continues below

Choice	A_follower_count	A_following_count	A_listed_count
Length:5500	Min. : 16	Min. : 0	Min. : 0
Class :character	1st Qu.: 2664	1st Qu.: 322	1st Qu.: 85
Mode :character	Median : 45589	Median : 778	Median : 932
NA	Mean : 649884	Mean : 12659	Mean : 5952
NA	3rd Qu.: 392738	3rd Qu.: 2838	3rd Qu.: 6734
NA	Max. :36543194	Max. :1165830	Max. :549144

Table 7: Table continues below

A_mentions_received	A_retweets_received	A_mentions_sent	A_retweets_sent
Min. : 0.1	Min. : 0.1	Min. : 0.1005	Min. : 0.1005
1st Qu.: 3.5	1st Qu.: 0.7	1st Qu.: 0.3595	1st Qu.: 0.1005
Median : 48.8	Median : 14.0	Median : 2.2997	Median : 0.3419
Mean : 2666.0	Mean : 1032.4	Mean : 6.0119	Mean : 1.1099
3rd Qu.: 349.8	3rd Qu.: 118.7	3rd Qu.: 7.1983	3rd Qu.: 1.3207
Max. :1145219.0	Max. :435825.9	Max. :76.8095	Max. :16.2905

Table 8: Table continues below

A_posts	A_network_feature_1	A_network_feature_2
Min. : 0.1005	Min. : 0	Min. : 0.00
1st Qu.: 0.6324	1st Qu.: 12	1st Qu.: 14.99
Median : 3.5552	Median : 195	Median : 54.93
Mean : 9.0907	Mean : 5268	Mean : 84.81
3rd Qu.: 10.6919	3rd Qu.: 1323	3rd Qu.: 109.70

A_posts	A_network_feature_1	A_network_feature_2
Max. :193.0724	Max. :920838	Max. :1121.00

Table 9: Table continues below

A_network_feature_3	B_follower_count	B_following_count	B_listed_count
Min. : 0 1st Qu.: 1181 Median : 2206 Mean : 3747 3rd Qu.: 4390 Max. :144651	Min. : 20 1st Qu.: 2498 Median : 44027 Mean : 685487 3rd Qu.: 370114 Max. :36543194	Min. : 0 1st Qu.: 322 Median : 773 Mean : 12738 3rd Qu.: 2838 Max. :664324	Min. : 0 1st Qu.: 75 Median : 890 Mean : 5903 3rd Qu.: 6734 Max. :549144

Table 10: Table continues below

B_mentions_received	B_retweets_received	B_mentions_sent	B_retweets_sent
Min. : 0.1 1st Qu.: 3.3 Median : 48.8 Mean : 2554.6 3rd Qu.: 374.4 Max. :1145219.0	Min. : 0.1 1st Qu.: 0.7 Median : 14.0 Mean : 997.1 3rd Qu.: 107.1 Max. :435825.9	Min. : 0.1005 1st Qu.: 0.3569 Median : 2.2514 Mean : 6.0997 3rd Qu.: 6.8668 Max. :76.8095	Min. : 0.1005 1st Qu.: 0.1005 Median : 0.3419 Mean : 1.1062 3rd Qu.: 1.3207 Max. :16.2905

Table 11: Table continues below

B_posts	B_network_feature_1	B_network_feature_2
Min. : 0.1005 1st Qu.: 0.8226 Median : 3.3430 Mean : 9.5058 3rd Qu.: 10.6005 Max. :193.0724	Min. : 0 1st Qu.: 11 Median : 190 Mean : 5255 3rd Qu.: 1323 Max. :920838	Min. : 0.00 1st Qu.: 15.18 Median : 54.93 Mean : 85.02 3rd Qu.: 112.19 Max. :1861.58

Table 12: Table continues below

B_network_feature_3	rapp_follower_count	rapp_following_count
Min. : 0 1st Qu.: 1206 Median : 2206 Mean : 3745 3rd Qu.: 4350 Max. :75526	Min. : 0.0 1st Qu.: 0.1 Median : 1.0 Mean : 609.1 3rd Qu.: 17.6 Max. :477141.0	Min. : 0.0 1st Qu.: 0.2 Median : 1.0 Mean : 253.5 3rd Qu.: 5.9 Max. :550744.0

Table 13: Table continues below

rapp_listed_count	rapp_mentions_received	rapp_retweets_received
Min. : 0.00	Min. : 0.0	Min. : 0.0
1st Qu.: 0.08	1st Qu.: 0.1	1st Qu.: 0.1
Median : 1.08	Median : 1.0	Median : 1.0
Mean : 192.26	Mean : 1406.2	Mean : 1223.3
3rd Qu.: 13.09	3rd Qu.: 19.8	3rd Qu.: 21.2
Max. :90405.00	Max. :1727666.0	Max. :520874.9

Table 14: Table continues below

rapp_mentions_sent	rapp_retweets_sent	rapp_posts
Min. : 0.0013	Min. : 0.00617	Min. : 0.0005
1st Qu.: 0.1626	1st Qu.: 0.27954	1st Qu.: 0.1701
Median : 1.0000	Median : 1.00000	Median : 1.0000
Mean : 14.6858	Mean : 5.58557	Mean : 14.1958
3rd Qu.: 6.2542	3rd Qu.: 3.62966	3rd Qu.: 5.5832
Max. :764.2482	Max. :137.97700	Max. :1921.0544

Table 15: Table continues below

rapp_network_feature_1	rapp_network_feature_2	rapp_network_feature_3
Min. : 0.00	Min. : 0.0000	Min. : 0.00
1st Qu.: 0.05	1st Qu.: 0.2409	1st Qu.: 0.34
Median : 1.00	Median : 1.0000	Median : 0.95
Mean : 610.66	Mean : 11.6571	Mean : 177.45
3rd Qu.: 18.57	3rd Qu.: 3.8490	3rd Qu.: 2.85
Max. :213718.00	Max. :1387.9091	Max. :44566.14

Table 16: Table continues below

A_foll_ratio	A_ment_ratio	A_retw_ratio	B_foll_ratio
Min. :0.000000	Min. : 0.000003	Min. : 0.000003	Min. :0.000000
1st Qu.:0.003375	1st Qu.: 0.008642	1st Qu.: 0.004788	1st Qu.:0.003339
Median :0.076572	Median : 0.056827	Median : 0.037888	Median :0.073903
Mean :0.379249	Mean : 0.334159	Mean : 0.398912	Mean :0.373494
3rd Qu.:0.534104	3rd Qu.: 0.313152	3rd Qu.: 0.328364	3rd Qu.:0.526048
Max. :9.190476	Max. :15.294775	Max. :11.157260	Max. :9.190476

Table 17: Table continues below

B_ment_ratio	B_retw_ratio	A_zeros	B_zeros
Min. : 0.000003	Min. : 0.000003	Min. :0.000	Min. :0.0000
1st Qu.: 0.008642	1st Qu.: 0.004788	1st Qu.:0.000	1st Qu.:0.0000
Median : 0.055929	Median : 0.038039	Median :0.000	Median :0.0000
Mean : 0.337523	Mean : 0.405542	Mean :0.166	Mean :0.1564
3rd Qu.: 0.302336	3rd Qu.: 0.328364	3rd Qu.:0.000	3rd Qu.:0.0000

B_ment_ratio	B_retw_ratio	A_zeros	B_zeros
Max. :13.255072	Max. :13.338818	Max. :5.000	Max. :5.0000

has_zeros
Mode :logical FALSE:4754 TRUE :746 NA NA NA

Selezione delle variabili tramite un albero di classificazione

```
set.seed(123)
metric <- "ROC"
Ctrl <- trainControl(method = "cv" , number=10, classProbs = TRUE,
                     summaryFunction = twoClassSummary)
rpartTune <- train(Choice ~ ., data = train, method = "rpart",
                   tuneLength = 15, trControl = Ctrl, metric=metric)
```

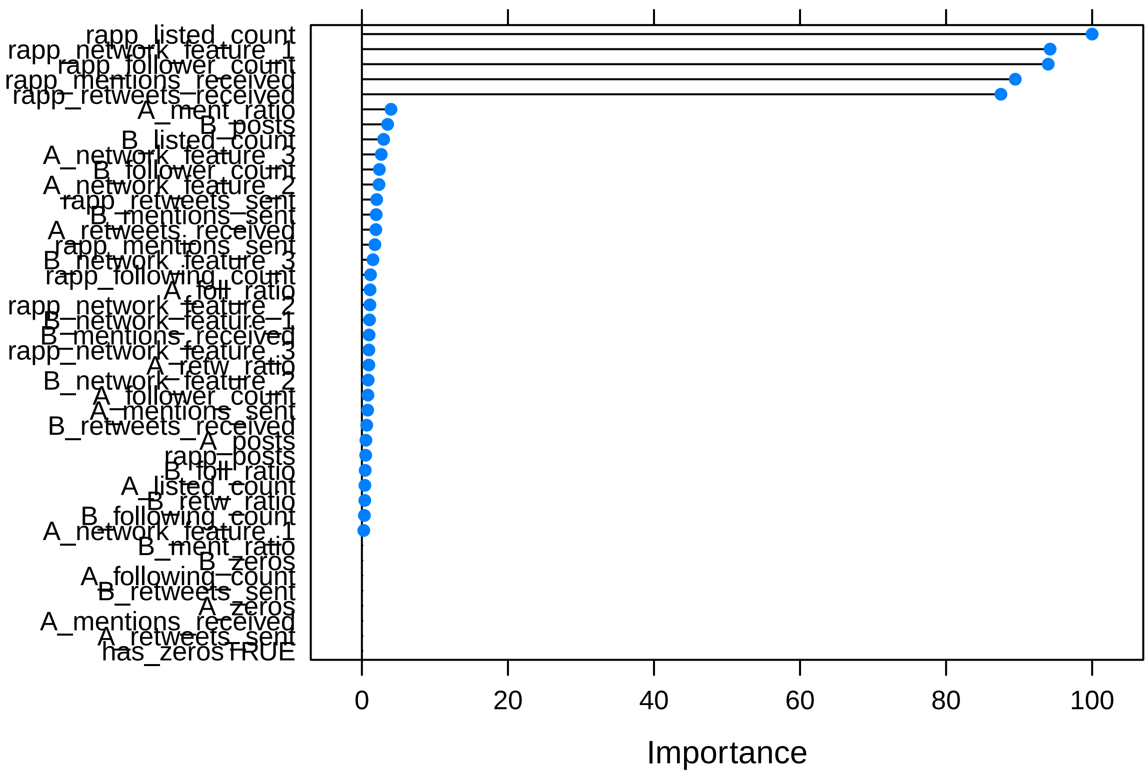
```
rpartTune
```

```
## CART
##
## 5500 samples
##   42 predictor
##    2 classes: 'A', 'B'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4950, 4951, 4950, 4951, 4950, 4950, ...
## Resampling results across tuning parameters:
##
##   cp            ROC        Sens       Spec
##   0.0009266123  0.8182915  0.7573399  0.7357552
##   0.0011119348  0.8216065  0.7591243  0.7405700
##   0.0012972572  0.8253175  0.7666154  0.7361256
##   0.0013899185  0.8247060  0.7626945  0.7390885
##   0.0014825797  0.8231596  0.7630427  0.7368567
##   0.0016679021  0.8263638  0.7580465  0.7453807
##   0.0017605634  0.8266267  0.7573322  0.7476112
##   0.0018532246  0.8250727  0.7566218  0.7498251
##   0.0020385471  0.8209134  0.7562621  0.7542696
##   0.0025945145  0.8163874  0.7612532  0.7568677
##   0.0033358043  0.8063396  0.7676906  0.7468759
##   0.0038917717  0.8015270  0.7726906  0.7468746
##   0.0044477391  0.8015495  0.7684024  0.7491119
##   0.0058067704  0.7702202  0.8048017  0.7142434
##   0.5207561156  0.6239966  0.9079931  0.3400000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.001760563.
```

```r
pander(getTrainPerf(rpartTune))
```

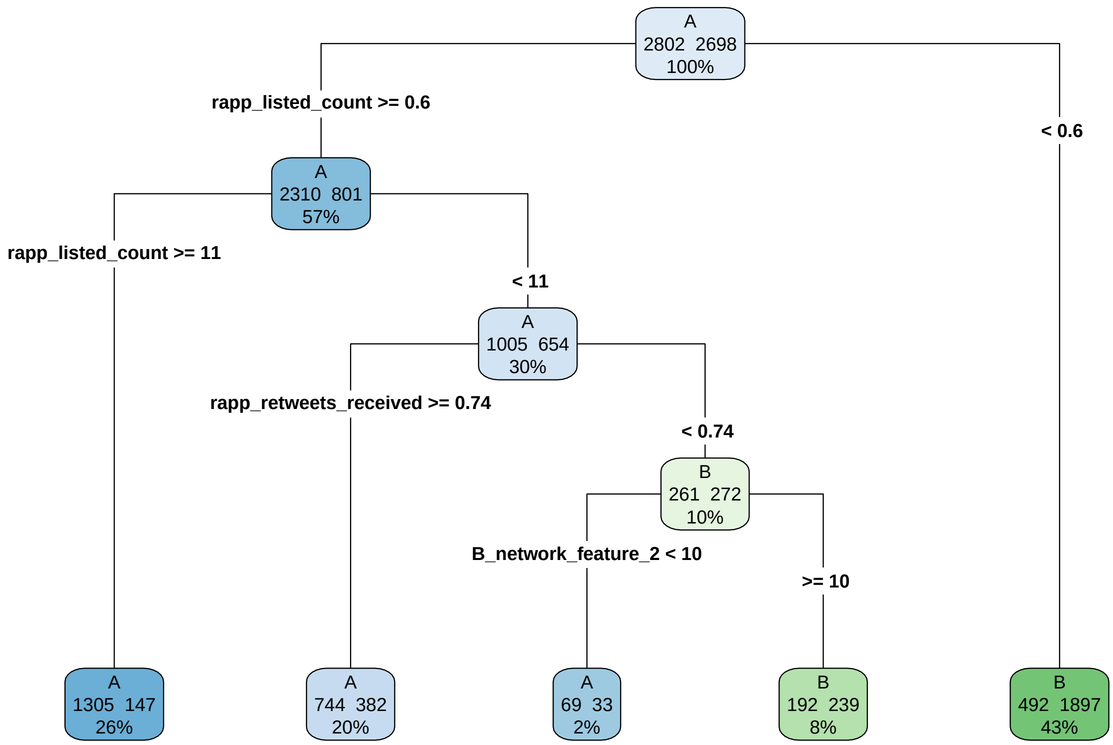
TrainROC	TrainSens	TrainSpec	method
0.8266	0.7573	0.7476	rpart

```r
Vimportance <- varImp(rpartTune)
plot(Vimportance)
```

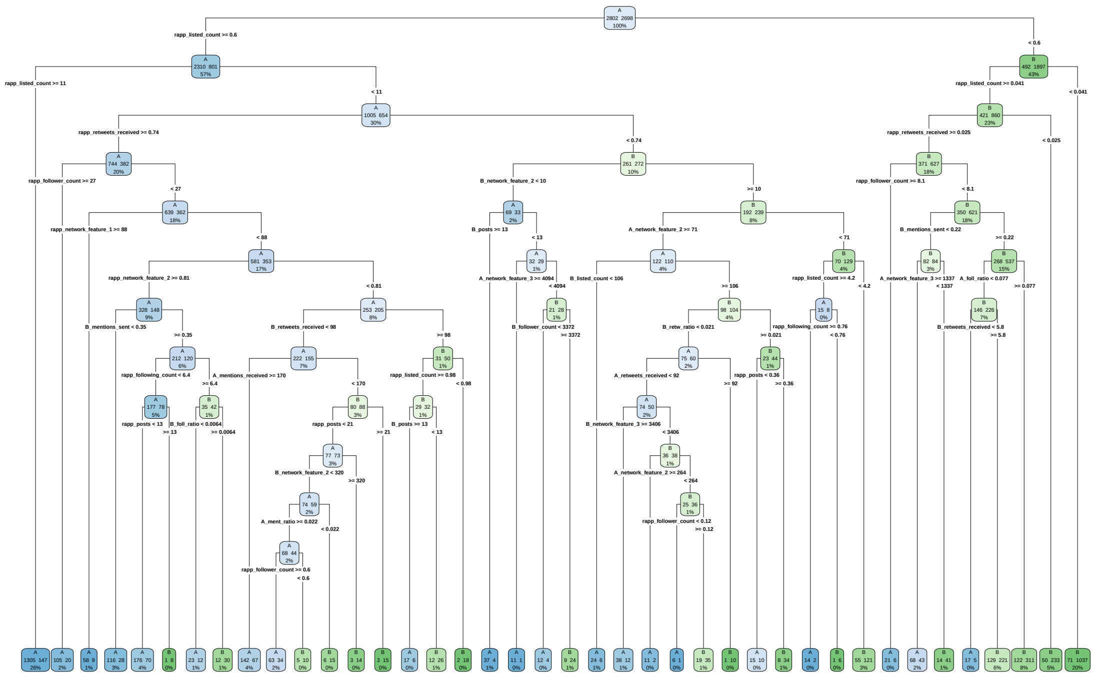



```r
set.seed(123)
Ctrl_save <- trainControl(method = "cv" , number=10, summaryFunction = twoClassSummary,
                          classProbs = TRUE, savePredictions = TRUE)
rpartTuneMy <- train(Choice ~ ., data = train, method = "rpart",
                     tuneGrid=data.frame(cp=0.0044477391),
                     trControl = Ctrl_save, metric=metric)
```

```r
set.seed(123)
mytree <- rpart(Choice ~ ., data = train, method = "class", cp = 0.0044477391)
rpart.plot(mytree, type = 4, extra = 101)
```



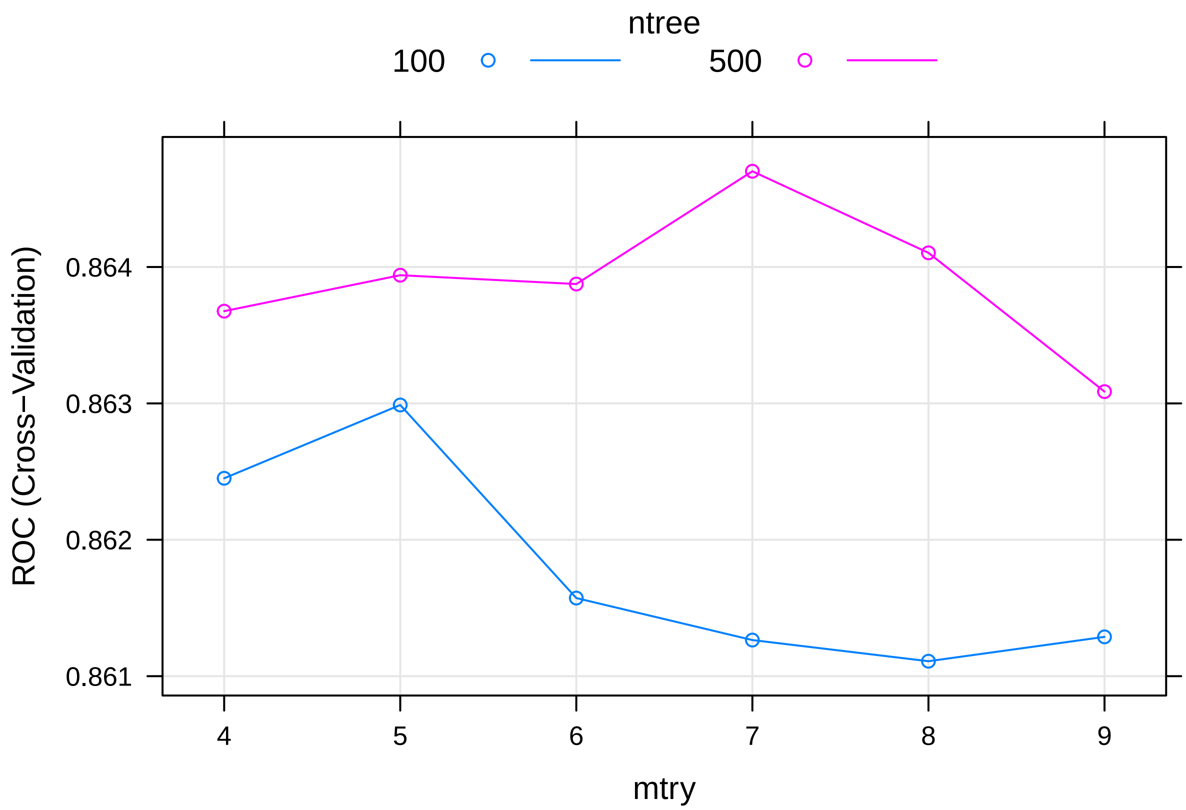

```r
set.seed(123)
mytree <- rpart(Choice ~ ., data = train, method = "class", cp = 0.001760563)
rpart.plot(mytree, type = 4, extra = 101)
```



Random Forest

```
customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"),
                                  class = rep("numeric", 2),
                                  label = c("mtry", "ntree"))
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes

set.seed(123)
tunegrid <- expand.grid(.mtry=c(4:9), .ntree=c(100,500))
rpartTuneMyRf <- train(Choice ~ ., data = train, method = customRF,
                       tuneGrid=tunegrid, trControl = Ctrl, metric=metric)
```

```
plot(rpartTuneMyRf)
```




```
set.seed(123)
tunegrid <- expand.grid(.mtry=7, .ntree=500)
rpartTuneMyRf_ok <- train(Choice ~ ., data = train, method = customRF,
                          tuneGrid=tunegrid, trControl = Ctrl_save, metric=metric)
```

XGBoost, tuning automatico e tramite griglia

```
set.seed(123)
fit.xgbTree.autoTune <- train(Choice ~ ., data=train, method="xgbTree",
                              metric=metric, trControl=Ctrl, tuneLength=3)
```

```
pander(fit.xgbTree.autoTune$bestTune)
```

Table 20: Table continues below

	nrounds	max_depth	eta	gamma	colsample_bytree
26	100	2	0.3	0	0.6

	min_child_weight	subsample
26	1	1

```
param = expand.grid(
  nrounds = seq(85,95,5),
  max_depth = 2,
```

```r
  eta = seq(0.25,0.35,0.05),
  gamma = seq(0.25,0.75,0.25),
  colsample_bytree = 0.5,
  min_child_weight = seq(0.5,1,0.5),
  subsample=1
)

set.seed(123)
fit.xgbTree <- train(Choice ~ ., data=train, method="xgbTree",
                     metric=metric, trControl=Ctrl, tuneGrid=param)
```

```r
pander(getTrainPerf(fit.xgbTree))
```

TrainROC	TrainSens	TrainSpec	method
0.8731	0.7952	0.7665	xgbTree

```r
param = expand.grid(
  nrounds = 90,
  max_depth = 2,
  eta = 0.25,
  gamma = 0.25,
  colsample_bytree = 0.5,
  min_child_weight = 0.5,
  subsample=1
)

set.seed(123)
fit.xgbTree.one.shot <- train(Choice ~ ., data=train, method="xgbTree",
                              metric=metric,trControl=Ctrl_save,tuneGrid=param)
```

Naive Bayes

```r
set.seed(123)
grid <- expand.grid(fL=0, usekernel = TRUE, adjust=1)
NBfit <- train(Choice ~ ., data = train, method="nb", tuneGrid=grid,
               trControl=Ctrl_save, metric=metric)
```

```r
pander(getTrainPerf(NBfit))
```

TrainROC	TrainSens	TrainSpec	method
0.8434	0.3002	0.9752	nb

Stochastic Gradient Boosting (il tuning automatico risulta essere il migliore)

```r
set.seed(123)
STGfit <- train(Choice ~ ., data = train, method="gbm", tuneLength=3,
                trControl=Ctrl, metric=metric)
```

```r
STGfit
```

```
## Stochastic Gradient Boosting
##
## 5500 samples
```

```
##   42 predictor
##    2 classes: 'A', 'B'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4950, 4951, 4950, 4951, 4950, 4950, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  ROC        Sens       Spec
##   1                   50      0.8608944  0.7808770  0.7565083
##   1                  100      0.8658719  0.7840913  0.7628074
##   1                  150      0.8674612  0.7869420  0.7624329
##   2                   50      0.8654140  0.7815951  0.7590913
##   2                  100      0.8699309  0.7901576  0.7672367
##   2                  150      0.8708453  0.7933668  0.7598293
##   3                   50      0.8670567  0.7894408  0.7620556
##   3                  100      0.8713576  0.7905071  0.7687319
##   3                  150      0.8720522  0.7883668  0.7683602
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
set.seed(123)
grid <- expand.grid(n.trees=150, interaction.depth=3, shrinkage=0.1, n.minobsinnode=10)
STGfit.one.shot <- train(Choice ~ ., data = train, method="gbm", tuneGrid=grid,
                trControl=Ctrl_save, metric=metric)
```

Per nnet e knn diamo in input le variabili sezionate tramite l'albero di classificazione, dato che l'inserimento di variabili non rilevanti potrebbe essere solo di disturbo

```r
TRAINSELECT2 <- train[, c(1,26,24, 27,28, 32)]
pander(summary(TRAINSELECT2))
```

Table 24: Table continues below

Choice	rapp_listed_count	rapp_follower_count
Length:5500	Min. : 0.00	Min. : 0.0
Class :character	1st Qu.: 0.08	1st Qu.: 0.1
Mode :character	Median : 1.08	Median : 1.0
NA	Mean : 192.26	Mean : 609.1
NA	3rd Qu.: 13.09	3rd Qu.: 17.6
NA	Max. :90405.00	Max. :477141.0

rapp_mentions_received	rapp_retweets_received	rapp_network_feature_1
Min. : 0.0	Min. : 0.0	Min. : 0.00
1st Qu.: 0.1	1st Qu.: 0.1	1st Qu.: 0.05
Median : 1.0	Median : 1.0	Median : 1.00
Mean : 1406.2	Mean : 1223.3	Mean : 610.66
3rd Qu.: 19.8	3rd Qu.: 21.2	3rd Qu.: 18.57

rapp_mentions_received	rapp_retweets_received	rapp_network_feature_1
Max. :1727666.0	Max. :520874.9	Max. :213718.00

Neural Network (preprocessing tramite pca, normalizzazione e standardizzazione)

```r
set.seed(123)
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003,  0.00001, 0.0001))
nnetFit_defgridDR1 <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                            method = "nnet",
                            preProcess = 'pca',
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 100)
```

```r
pander(getTrainPerf(nnetFit_defgridDR1))
```

TrainROC	TrainSens	TrainSpec	method
0.8571	0.6788	0.8354	nnet

```r
pander(nnetFit_defgridDR1$bestTune)
```

	size	decay
4	1	3e-04

```r
set.seed(123)
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003,  0.00001, 0.0001))

nnetFit_defgridDR3 <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                            method = "nnet",
                            preProcess = c('range'),
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 100)
```

```r
pander(getTrainPerf(nnetFit_defgridDR3))
```

TrainROC	TrainSens	TrainSpec	method
0.8577	0.4475	0.9448	nnet

```r
pander(nnetFit_defgridDR3$bestTune)
```

	size	decay
11	3	2e-04

```r
set.seed(123)
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003,  0.00001, 0.0001))
nnetFit_defgridDR2 <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                            method = "nnet",
                            preProcess = c('center', "scale"),
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 100)
```

```r
pander(getTrainPerf(nnetFit_defgridDR2))
```

TrainROC	TrainSens	TrainSpec	method
0.8581	0.697	0.8258	nnet

```r
pander(nnetFit_defgridDR2$bestTune)
```

	size	decay
7	2	2e-04

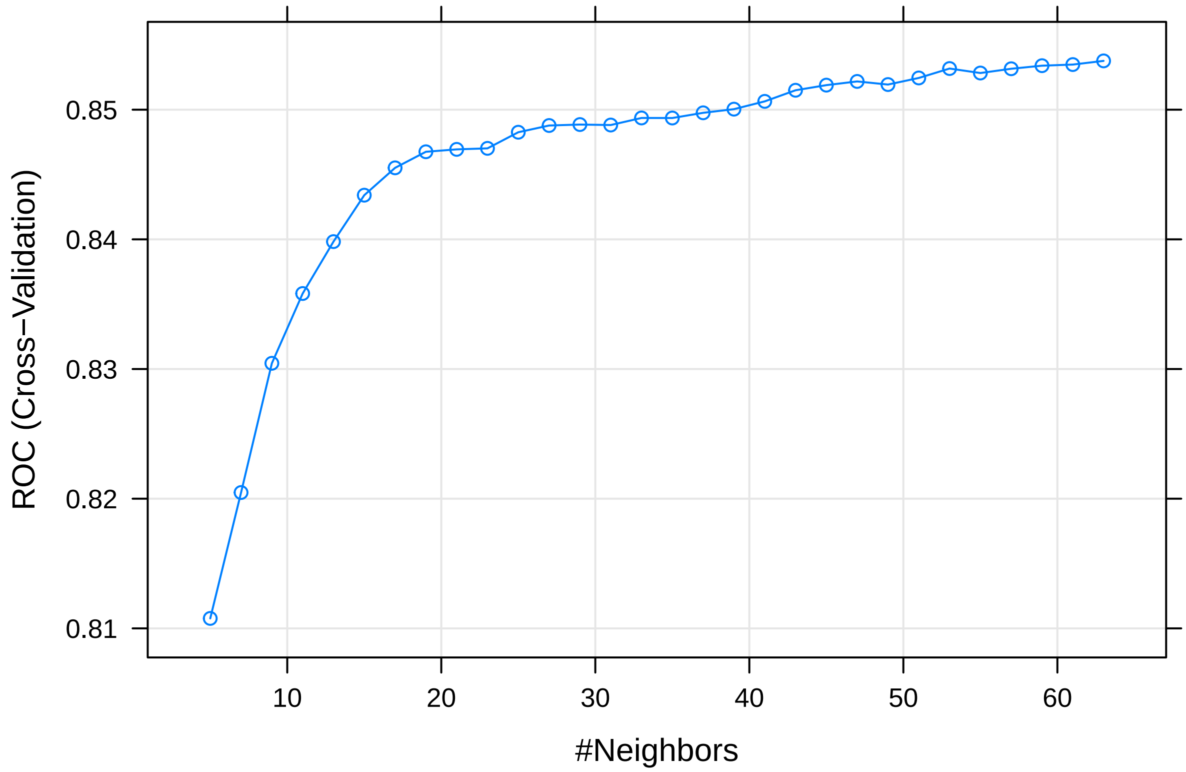
La migliore è quella con standardizzazione

```r
set.seed(123)
tunegrid <- expand.grid(size=2, decay =0.0002)
nnetFit_finale <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                         method = "nnet",
                         preProcess = c( 'center' ,'scale'),
                         metric=metric,
                         trControl=Ctrl_save, tuneGrid=tunegrid,
                         trace = FALSE,
                         maxit = 100)
```

K-Nearest Neighbors

```r
set.seed(123)
knnFit3 <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                 method = "knn", tuneLength = 30,
                 preProcess = c("center", "scale"),
                 metric=metric,
                 trControl = Ctrl)
```

```r
plot(knnFit3)
```

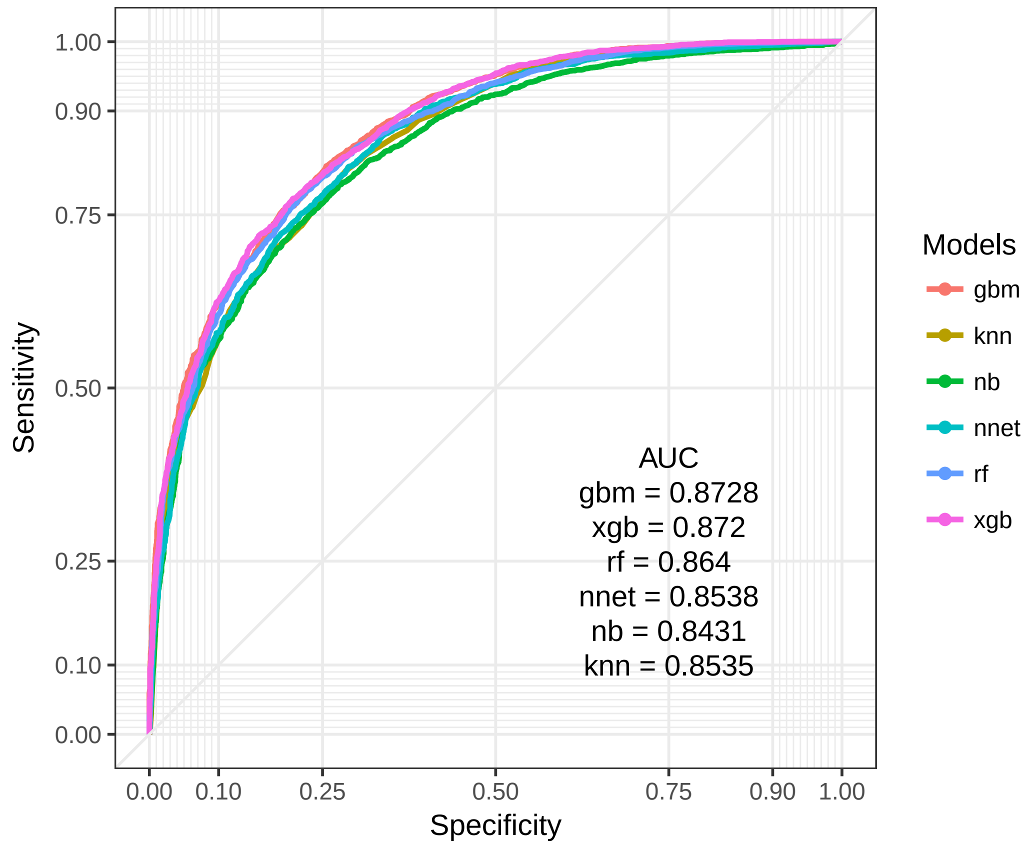



```
set.seed(123)
tunegrid <- expand.grid(k=63)
knnFit1f <- train(TRAINSELECT2[-1], TRAINSELECT2$Choice,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  metric=metric,tuneGrid=tunegrid,
                  trControl = Ctrl_save)
```

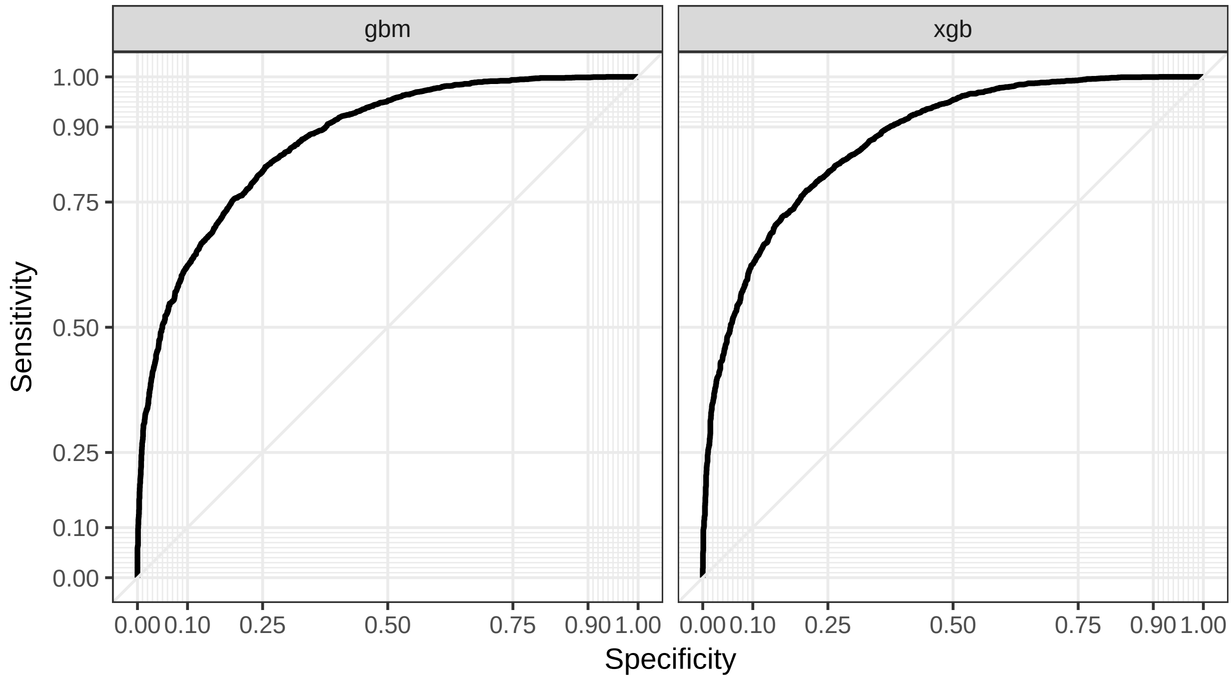
ROC curves

```
roc_values <- cbind(as.data.frame(fit.xgbTree.one.shot$pred$obs),
                    as.data.frame(fit.xgbTree.one.shot$pred$A))
gbm <- as.data.frame(STGfit.one.shot$pred$A)
nb <- as.data.frame(NBfit$pred$A)
nnet <- as.data.frame(nnetFit_finale$pred$A)
knn <- as.data.frame(knnFit1f$pred$A)
rf <- as.data.frame(rpartTuneMyRf_ok$pred$A)

roc_values <- cbind(roc_values, gbm, nb, nnet, knn, rf)
names(roc_values) <- c("obs","xgb","gbm", "nb", "nnet", "knn", "rf")

longtest <- melt_roc(roc_values, "obs", c("xgb", "gbm", "nb", "nnet", "knn", "rf"))
longtest$D <- ifelse(longtest$D=="A",1,0)
names(longtest)[3] <- "Models"

g <- ggplot(longtest, aes(m=M, d=D, color=Models)) +
  geom_roc(n.cuts=0) +
```

```r
  coord_equal() +
  style_roc(xlab="Specificity", ylab="Sensitivity")

g + annotate("text", x=0.75, y=0.4, label="AUC") +
  annotate("text", x=0.75, y=0.35, label=paste("gbm =", round(calc_auc(g)$AUC[1], 4))) +
  annotate("text", x=0.75, y=0.30, label=paste("xgb =", round(calc_auc(g)$AUC[6], 4))) +
  annotate("text", x=0.75, y=0.25, label=paste("rf =", round(calc_auc(g)$AUC[5], 4))) +
  annotate("text", x=0.75, y=0.20, label=paste("nnet =", round(calc_auc(g)$AUC[4], 4))) +
  annotate("text", x=0.75, y=0.15, label=paste("nb =", round(calc_auc(g)$AUC[3], 4))) +
  annotate("text", x=0.75, y=0.10, label=paste("knn =", round(calc_auc(g)$AUC[2], 4)))
```




```r
roc_best <- roc_values[,c("obs","xgb","gbm")]

longtest2 <- melt_roc(roc_best, "obs", c("xgb", "gbm"))
longtest2$D <- ifelse(longtest2$D=="A",1,0)
names(longtest2)[3] <- "Models"

ggplot(longtest2, aes(m=M, d=D)) +
  geom_roc(n.cuts=0) +
  coord_equal() +
  facet_wrap(~Models) +
  style_roc(xlab="Specificity", ylab="Sensitivity")
```



Confusion matrix

```
confusionMatrix(fit.xgbTree.one.shot)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B
##          A 40.4 11.3
##          B 10.6 37.8
##
##  Accuracy (average) : 0.7815
```

Codice per il test set (da vedere la performance su kaggle)

```
test <- read.csv("~/data_science_lab/test.csv")

for (i in (1:11)){
  test[,(i+22)] <- test[,i]/test[,(11+i)]
  names(test)[i+22] <- paste("rapp", substring(names(test)[i], 2), sep="")
}

for (i in (23:ncol(test))){
  test[is.na(test[,i]),i] <- 1
  test[test[,i]==Inf,i] <- test[test[,i]==Inf,(i-22)]
}
```

```r
test$A_foll_ratio <- test$A_following_count/test$A_follower_count
test$A_ment_ratio <- test$A_mentions_sent/test$A_mentions_received
test$A_retw_ratio <- test$A_retweets_sent/test$A_retweets_received

test$B_foll_ratio <- test$B_following_count/test$B_follower_count
test$B_ment_ratio <- test$B_mentions_sent/test$B_mentions_received
test$B_retw_ratio <- test$B_retweets_sent/test$B_retweets_received

test$A_zeros <- 0
test$B_zeros <- 0
for (i in (1:11)){
  test$A_zeros[test[,i]==0] <- test$A_zeros[test[,i]==0] + 1
}
for (i in (12:22)){
  test$B_zeros[test[,i]==0] <- test$B_zeros[test[,i]==0] + 1
}

test$has_zeros <- FALSE
test$has_zeros[(test$A_zeros+test$B_zeros)>0]<-TRUE
```

```r
preds=predict(fit.xgbTree.one.shot, newdata = test, type="prob", digits=9)
pander(head(preds))
```

A	B
0.221	0.779
0.522	0.478
0.04949	0.9505
0.1718	0.8282
0.5292	0.4708
0.3193	0.6807

```r
submission <- as.data.frame(cbind(c(1:nrow(test)), preds[,"A"]))
names(submission)<-c("Id","Choice")
pander(head(submission))
```

Id	Choice
1	0.221
2	0.522
3	0.04949
4	0.1718
5	0.5292
6	0.3193

```r
write.csv(submission, file="mysub.csv", row.names=FALSE)
```

GBM & XGB Scores

Submission and Description	Private Score	Public Score	Use for Final Score
gbm.csv just now by Alex Ceccotti add submission details	0.86613	0.86457	☐
xgb.csv 2 minutes ago by Alex Ceccotti add submission details	0.86568	0.86677	☐

Public Leaderboard

34	new	Van Zeidt	0.86681	11	5y
35	new	Jure Zbontar	0.86678	2	5y
36	new	vyatka	0.86656	8	5y
37	new	dh_weekenders	0.86605	57	5y

Private Leaderboard

45	▾3	ziyuang	0.86617	23	5y
46	▾28	The Classy Fires	0.86617	32	5y
47	▾8	Matt Sco	0.86605	12	5y
48	▾8	Tony_R	0.86568	11	5y

Figure 1: Results