Ensemble_Learning_MU3D

2022-11-03

load packages

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.2.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Warning: package 'stringr' was built under R version 4.2.2

## Warning: package 'forcats' was built under R version 4.2.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.2.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.2.2

##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.2.2

## corrplot 0.92 loaded
```

```r
library(FactoMineR)
```

```
## Warning: package 'FactoMineR' was built under R version 4.2.2
```

```r
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.2.2

## Loading required package: usethis

## Warning: package 'usethis' was built under R version 4.2.2
```

```r
install_github('sinhrks/ggfortify',force = TRUE)
```

```
## WARNING: Rtools is required to build R packages, but is not currently installed.
##
## Please download and install Rtools 4.2 from https://cran.r-project.org/bin/windows/Rtools/ or https:/
## Downloading GitHub repo sinhrks/ggfortify@HEAD

## WARNING: Rtools is required to build R packages, but is not currently installed.
##
## Please download and install Rtools 4.2 from https://cran.r-project.org/bin/windows/Rtools/ or https:/

##          checking for file 'C:\Users\kunbu\AppData\Local\Temp\RtmpaQeabl\remotes216c62736c34\sinhrks
##       -  preparing 'ggfortify':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  chec
##       -  checking for LF line-endings in source and make files and shell scripts
##       -  checking for empty or unneeded directories
##       -  building 'ggfortify_0.4.15.tar.gz'
##
##
```

```r
library(ggfortify)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.2.2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.2
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:psych':
##
##     outlier
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

import video level dataset

Boxplot for variables

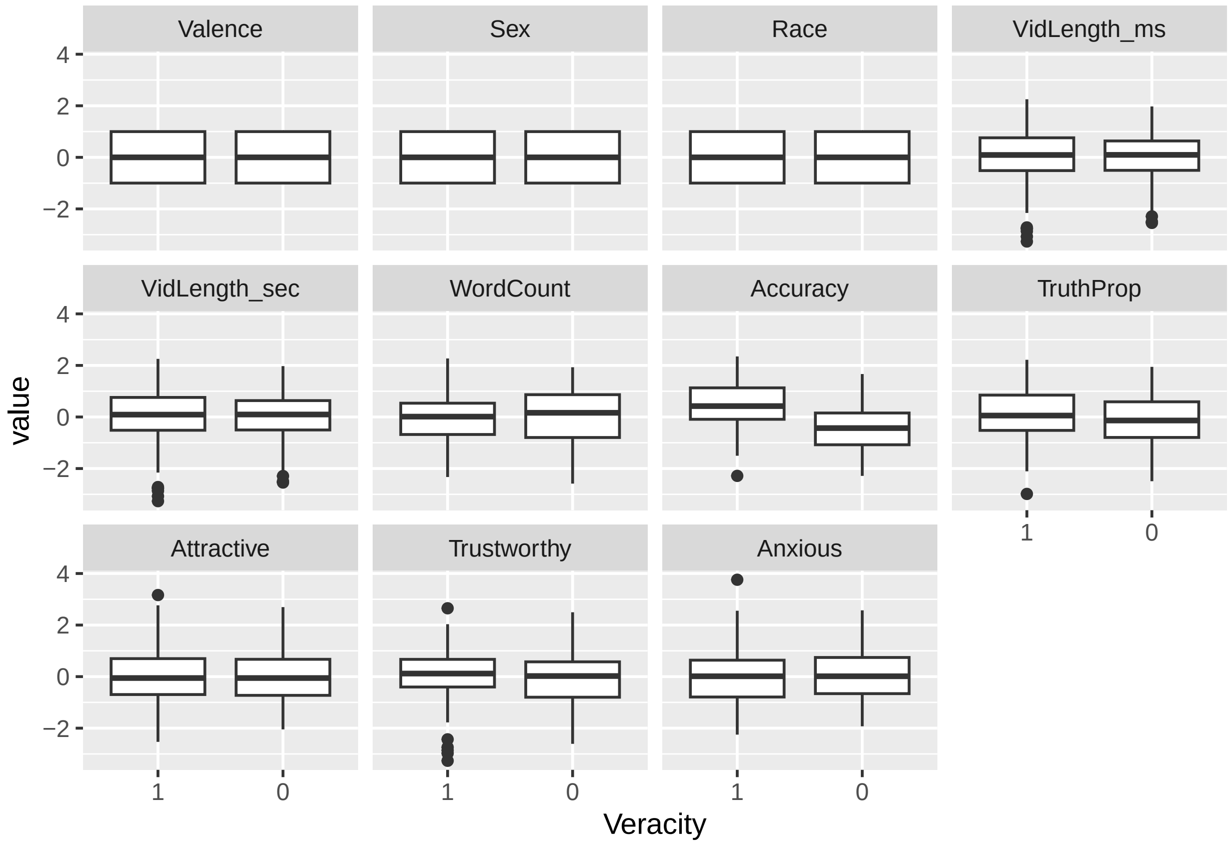
```
#remove veractiy first
MU3D_Video_Level_Data <- MU3D_Video_Level_Data0[,-grep("Veracity",colnames(MU3D_Video_Level_Data0))]
str(MU3D_Video_Level_Data)
```

```
## 'data.frame':    320 obs. of  13 variables:
##  $ VideoID      : chr  "BF001_1PT" "BF001_2NL" "BF001_3NT" "BF001_4PL" ...
##  $ Valence      : int  1 0 0 1 1 0 0 1 1 0 ...
##  $ Sex          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Race         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ VidLength_ms : int  38783 37120 38484 38026 36351 36650 29141 36480 36960 29610 ...
##  $ VidLength_sec: num  38.8 37.1 38.5 38 36.4 ...
##  $ WordCount    : int  110 88 120 124 91 73 95 104 91 114 ...
##  $ Accuracy     : num  0.77 0.4 0.77 0.58 0.59 0.33 0.6 0.64 0.64 0.27 ...
##  $ TruthProp    : num  0.77 0.6 0.77 0.42 0.59 0.67 0.6 0.36 0.64 0.73 ...
##  $ Attractive   : num  4.55 3.55 3.27 4.05 4.86 5.05 4.4 4.27 3.09 3 ...
##  $ Trustworthy  : num  4.32 3.75 3.95 4.05 4.36 4.62 4.5 3.73 4.27 4.55 ...
##  $ Anxious      : num  3.18 3.05 2.82 3.11 3.32 2.33 3.15 2.91 2.64 2.73 ...
##  $ Transcription: chr  "My best friend is a really nice person. Um. She's always kind to everyone. Sh
```

```r
#scaled
colnames(MU3D_Video_Level_Data)
```

```
##  [1] "VideoID"       "Valence"       "Sex"           "Race"
##  [5] "VidLength_ms"  "VidLength_sec" "WordCount"     "Accuracy"
##  [9] "TruthProp"     "Attractive"    "Trustworthy"   "Anxious"
## [13] "Transcription"
```

```r
MU3D_Video_Level_Data.scaled <- data.frame(scale(MU3D_Video_Level_Data[,-grep("VideoID|Transcription",co

#level veractiy
levels0 <- unique(c(MU3D_Video_Level_Data0$Veracity, MU3D_Video_Level_Data0$Veracity))
#add veracity back
MU3D_Video_Level_Data.scaled$Veracity <- factor(MU3D_Video_Level_Data0$Veracity,levels = levels0)

#melt data for boxplot
MU3D_Video_Level_Data.scaled.melt <- melt(MU3D_Video_Level_Data.scaled, id.var = "Veracity")

#plot boxplot
ggplot(data = MU3D_Video_Level_Data.scaled.melt, aes(x=Veracity, y = value))+
  geom_boxplot() +
  facet_wrap(~variable, ncol=4)
```



Feature Selection

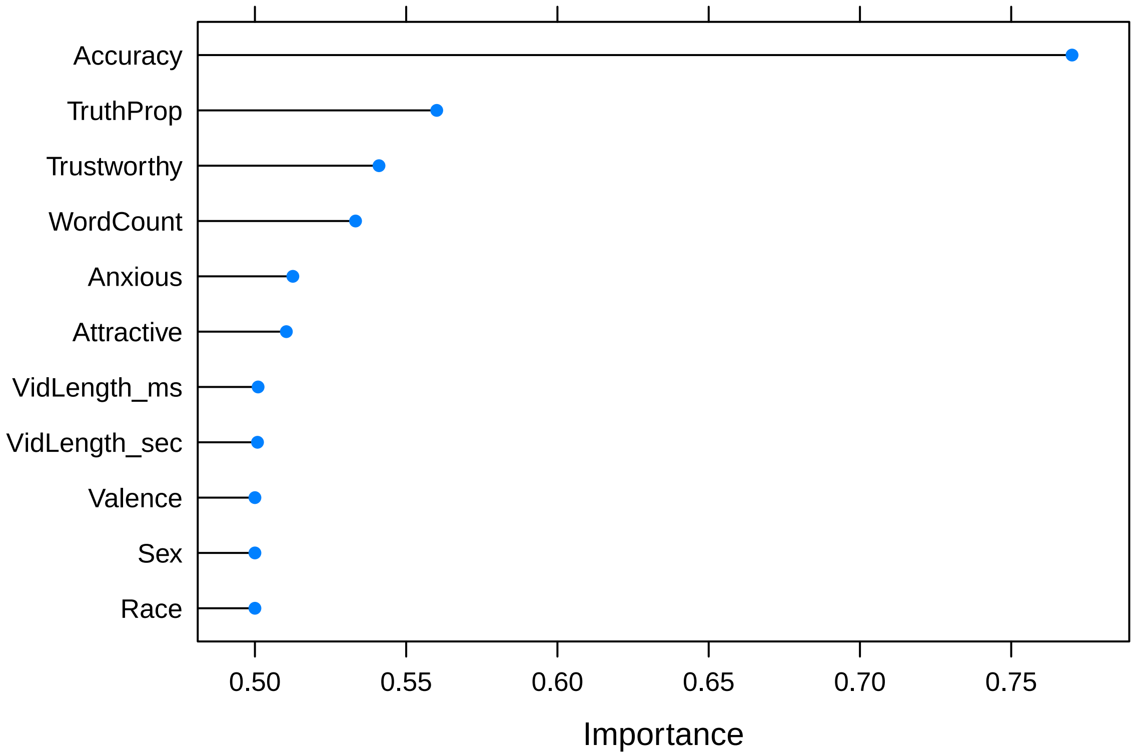
```r
# ensure results are repeatable
set.seed(7)
# load the library
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.2.2
```

```r
library(caret)

# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(Veracity~., data=MU3D_Video_Level_Data.scaled, method="lvq", preProcess="scale", trContr
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## ROC curve variable importance
##
##               Importance
## Accuracy          0.7701
## TruthProp         0.5601
## Trustworthy       0.5410
## WordCount         0.5333
## Anxious           0.5125
## Attractive        0.5104
## VidLength_ms      0.5011
## VidLength_sec     0.5009
## Sex               0.5000
## Valence           0.5000
## Race              0.5000
```

```r
# plot importance
plot(importance)
```




```
features <- colnames(MU3D_Video_Level_Data.scaled[,importance$importance$X0>=0.51])
```

Individual learning classifer

Splitting data set into training and test datasets using 80/20 cretira.

```
#split train and test 80/20
set.seed(123)
smp_size_raw <- floor(0.80 * nrow(MU3D_Video_Level_Data.scaled))
train_ind_raw <- sample(nrow(MU3D_Video_Level_Data.scaled), size = smp_size_raw)
train_raw.df <- as.data.frame(MU3D_Video_Level_Data.scaled[train_ind_raw, importance$importance$X0>=0.5
test_raw.df <- as.data.frame(MU3D_Video_Level_Data.scaled[-train_ind_raw, importance$importance$X0>=0.5
train_raw.df$Veracity <- MU3D_Video_Level_Data.scaled[train_ind_raw, 12]
test_raw.df$Veracity <- MU3D_Video_Level_Data.scaled[-train_ind_raw, 12]
```

SVM

```
poly.tune <- tune.svm(Veracity ~ ., data = train_raw.df,
                      kernel = "polynomial",
                      degree = c(2, 3, 4, 5, 6),
                      coef0 = c(0.1, 0.5, 1, 2, 3, 4))
summary(poly.tune) #best degree is 4,coef0 = 2,  misclassification rate no larger than 12.87%
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       4     4
##
## - best performance: 0.1212308
##
## - Detailed performance results:
##    degree coef0     error dispersion
## 1       2   0.1 0.2187692 0.08477778
## 2       3   0.1 0.2190769 0.07740091
## 3       4   0.1 0.2809231 0.08762541
## 4       5   0.1 0.3198462 0.09631050
## 5       6   0.1 0.3393846 0.09094493
## 6       2   0.5 0.2153846 0.08745958
## 7       3   0.5 0.1923077 0.08548889
## 8       4   0.5 0.1924615 0.08580960
## 9       5   0.5 0.1884615 0.08634975
## 10      6   0.5 0.1884615 0.09945611
## 11      2   1.0 0.2153846 0.09278226
## 12      3   1.0 0.1607692 0.06875716
## 13      4   1.0 0.1804615 0.08473511
## 14      5   1.0 0.1607692 0.07611974
## 15      6   1.0 0.1524615 0.09130639
## 16      2   2.0 0.2153846 0.10125316
## 17      3   2.0 0.1569231 0.07513827
## 18      4   2.0 0.1333846 0.08342141
## 19      5   2.0 0.1444615 0.08240134
## 20      6   2.0 0.1447692 0.08342267
## 21      2   3.0 0.2113846 0.08681432
## 22      3   3.0 0.1607692 0.07319963
## 23      4   3.0 0.1252308 0.08762841
## 24      5   3.0 0.1567692 0.10907886
## 25      6   3.0 0.1413846 0.10289805
## 26      2   4.0 0.2152308 0.08481081
## 27      3   4.0 0.1489231 0.06117328
## 28      4   4.0 0.1212308 0.09574182
## 29      5   4.0 0.1449231 0.10847530
## 30      6   4.0 0.1295385 0.10254051
```

```r
best.poly <- poly.tune$best.model
poly.test <- predict(best.poly, newdata = test_raw.df)
table(poly.test, test_raw.df$Veracity)
```

```
##
## poly.test  1  0
##         1 27  8
##         0  2 27
```

```r
confusionMatrix(poly.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference"))
```

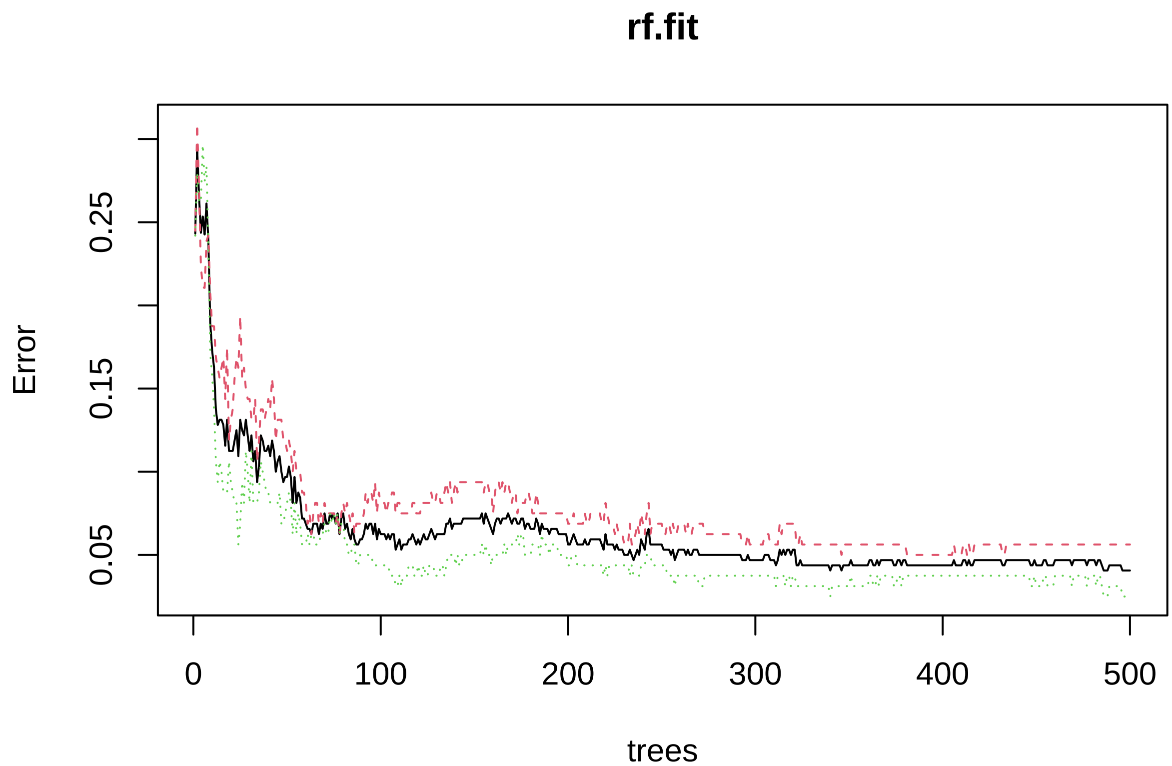
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1 27  8
##          0  2 27
##
##                Accuracy : 0.8438
##                  95% CI : (0.7314, 0.9224)
##     No Information Rate : 0.5469
##     P-Value [Acc > NIR] : 4.925e-07
##
##                   Kappa : 0.6902
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9310
##             Specificity : 0.7714
##          Pos Pred Value : 0.7714
##          Neg Pred Value : 0.9310
##              Prevalence : 0.4531
##          Detection Rate : 0.4219
##    Detection Prevalence : 0.5469
##       Balanced Accuracy : 0.8512
##
##        'Positive' Class : 1
##
```

Random Forest

```r
set.seed(123)
rf.fit <- randomForest(Veracity~., data= MU3D_Video_Level_Data.scaled)
rf.fit
```

```
##
## Call:
##  randomForest(formula = Veracity ~ ., data = MU3D_Video_Level_Data.scaled)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.06%
## Confusion matrix:
##     1   0 class.error
## 1 151   9     0.05625
## 0   4 156     0.02500
```

```
plot(rf.fit)
```

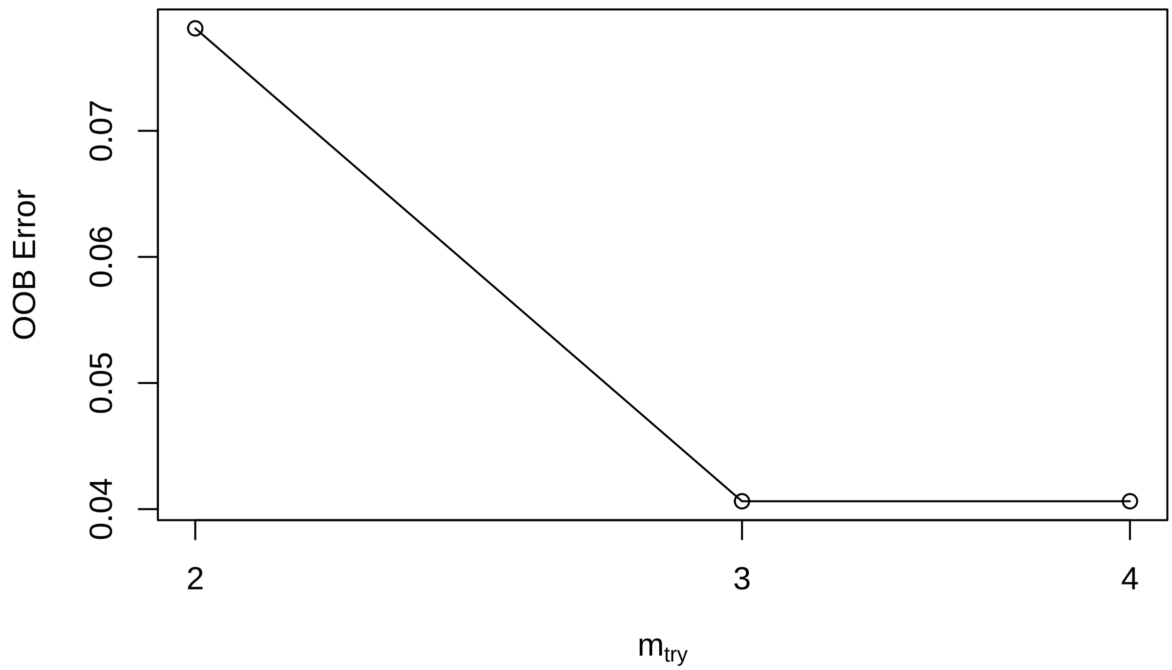



```
which.min(rf.fit$err.rate[,1])
```

```
## [1] 340
```

```
set.seed(123)
mtry <- tuneRF(MU3D_Video_Level_Data.scaled[,-12],MU3D_Video_Level_Data.scaled$Veracity, ntreeTry=1000,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 3  OOB error = 4.06%
## Searching left ...
## mtry = 2     OOB error = 7.81%
## -0.9230769 0.01
## Searching right ...
## mtry = 4     OOB error = 4.06%
## 0 0.01
```




```
best.m <- mtry[mtry[,2] == min(mtry[,2]), 1]
print(mtry)
```

```
##       mtry OOBError
## 2.OOB    2 0.078125
## 3.OOB    3 0.040625
## 4.OOB    4 0.040625
```

```
print(best.m)
```

```
## 3.OOB 4.OOB
##     3     4
```

```
set.seed(123)
rf.fit1 <-randomForest(Veracity~.,data=train_raw.df[,-3], mtry=best.m, importance=TRUE,ntree=334)
```

```
## Warning in mtry < 1 || mtry > p: 'length(x) = 2 > 1' in coercion to 'logical(1)'

## Warning in mtry < 1 || mtry > p: 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
print(rf.fit1)
```

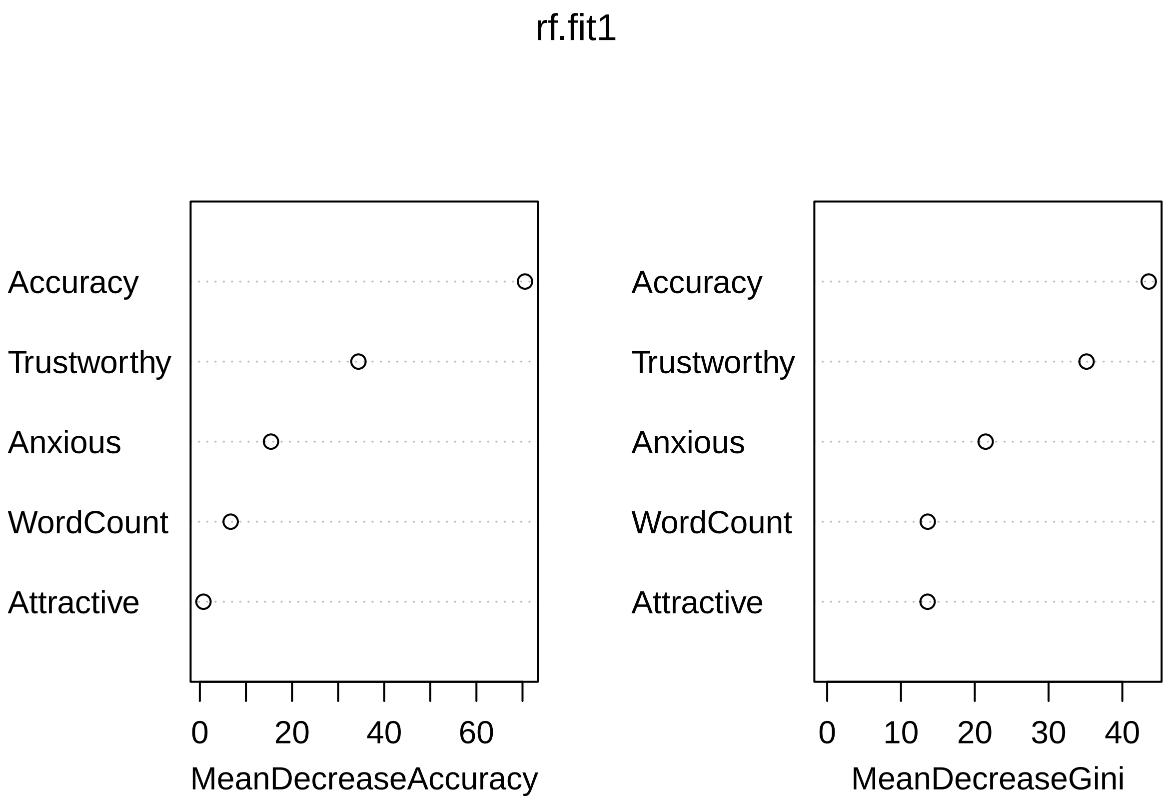
```
##
```

```
## Call:
##  randomForest(formula = Veracity ~ ., data = train_raw.df[, -3],      mtry = best.m, importance = TRU
##                Type of random forest: classification
##                      Number of trees: 334
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.05%
## Confusion matrix:
##    1  0 class.error
## 1 98 33   0.2519084
## 0 26 99   0.2080000
```

```
#Evaluate variable importance
importance(rf.fit1)
```

```
##                    1         0 MeanDecreaseAccuracy MeanDecreaseGini
## WordCount    5.469427  4.019675            6.6861805         13.62865
## Accuracy    51.273047 55.159668           70.5412102         43.57648
## Attractive  -1.342833  2.496349            0.7810218         13.60494
## Trustworthy 24.495995 27.475946           34.4083459         35.15233
## Anxious      6.050911 14.959367           15.4323272         21.47542
```

```
varImpPlot(rf.fit1)
```



```r
rf.pred <- predict(rf.fit1, test_raw.df)
confusionMatrix(rf.pred, test_raw.df$Veracity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1 22  7
##          0  7 28
##
##                Accuracy : 0.7812
##                  95% CI : (0.6603, 0.8749)
##     No Information Rate : 0.5469
##     P-Value [Acc > NIR] : 8.469e-05
##
##                   Kappa : 0.5586
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7586
##             Specificity : 0.8000
##          Pos Pred Value : 0.7586
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4531
##          Detection Rate : 0.3438
##    Detection Prevalence : 0.4531
##       Balanced Accuracy : 0.7793
##
##        'Positive' Class : 1
##
```

```r
set.seed(123)
pred1=predict(rf.fit1, test_raw.df, type = "prob")
library(ROCR)
```

plot AUC

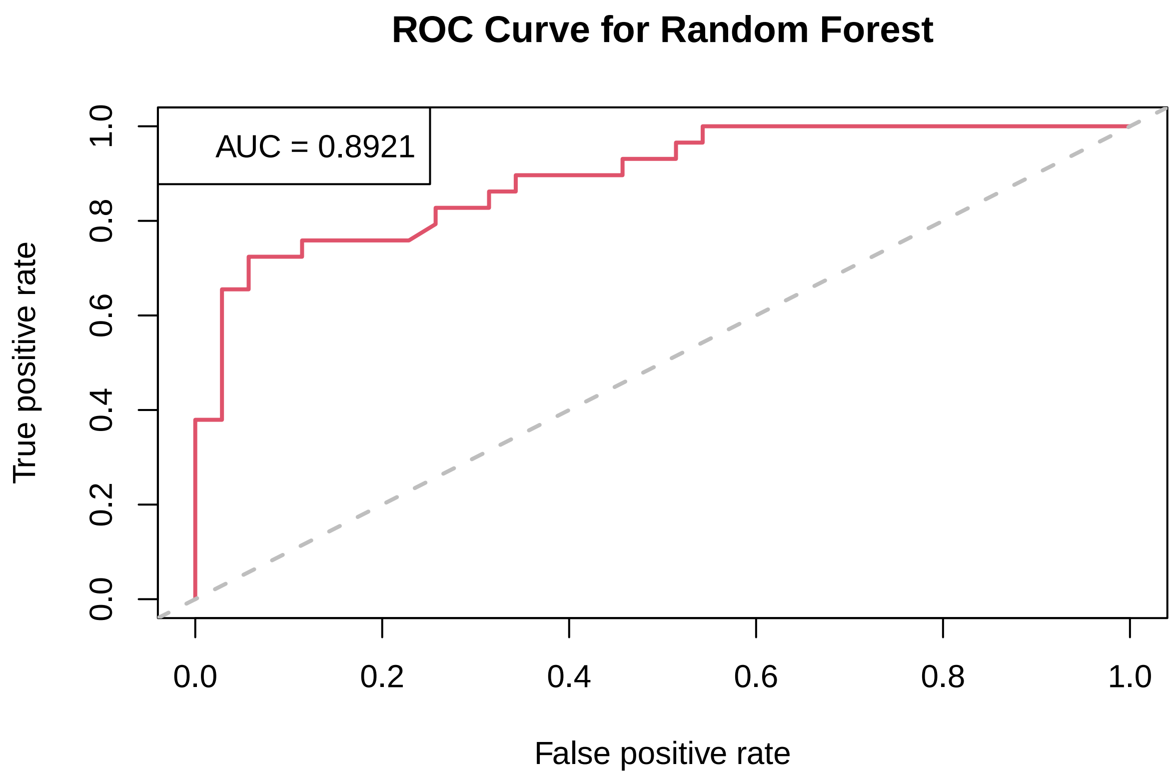
```
## Warning: package 'ROCR' was built under R version 4.2.2
```

```r
perf = prediction(pred1[,1], test_raw.df$Veracity)
# 1. Area under curve
auc = performance(perf, "auc")
auc@y.values[[1]]
```

```
## [1] 0.8921182
```

```r
# 2. True Positive and Negative Rate
pred3 = performance(perf, "tpr","fpr")
# 3. Plot the ROC curve
```

```r
plot(pred3,main="ROC Curve for Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
legend("topleft", c(paste0("AUC = ", round(auc@y.values[[1]],4))))
```



KNN

```r
library(class)

target_category <- train_raw.df$Veracity
test_category <- test_raw.df$Veracity
k=sqrt(dim(MU3D_Video_Level_Data)[1])
##run knn function
knn.fit <- knn(train_raw.df,test_raw.df,cl=target_category,k=k)

##create confusion matrix
confusionMatrix(knn.fit, test_raw.df$Veracity, dnn = c("Prediction", "Reference"))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  1  0
##            1 29  0
##            0  0 35
```

```
##
##                Accuracy : 1
##                  95% CI : (0.944, 1)
##     No Information Rate : 0.5469
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4531
##          Detection Rate : 0.4531
##    Detection Prevalence : 0.4531
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 1
##
```

GLM

```r
##run glm function
glm.fit <- glm(Veracity~. ,family = binomial(link = "logit"), train_raw.df,)
outcome <- predict(glm.fit, newdata = test_raw.df, type = 'response')
outcome1 <- as.factor(ifelse(outcome > 0.5, 1, 0))
##create confusion matrix
confusionMatrix(data = outcome1, test_raw.df$Veracity, dnn = c("Prediction", "Reference"))
```

```
## Warning in confusionMatrix.default(data = outcome1, test_raw.df$Veracity, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1  9 23
##          0 20 12
##
##                Accuracy : 0.3281
##                  95% CI : (0.2159, 0.4569)
##     No Information Rate : 0.5469
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : -0.3438
##
##  Mcnemar's Test P-Value : 0.7604
##
```

```
##             Sensitivity : 0.3103
##             Specificity : 0.3429
##          Pos Pred Value : 0.2812
##          Neg Pred Value : 0.3750
##              Prevalence : 0.4531
##          Detection Rate : 0.1406
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.3266
##
##        'Positive' Class : 1
##
```

WSRF

```
#install.packages("wsrf")
library(wsrf)
```

```
## Warning: package 'wsrf' was built under R version 4.2.2

## Loading required package: parallel

## Loading required package: Rcpp

## wsrf: An R Package for Scalable Weighted Subspace Random Forests.

## Version 1.7.27

## Use C++ standard thread library for parallel computing

##
## Attaching package: 'wsrf'

## The following objects are masked from 'package:randomForest':
##
##     combine, importance

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
target <- "Veracity"
ds <- MU3D_Video_Level_Data.scaled
vars <- names(ds)

if (sum(is.na(ds[vars]))) ds[vars] <- na.roughfix(ds[vars])
ds[target] <- as.factor(ds[[target]])
(tt <- table(ds[target]))
```

```
## Veracity
##   1   0
## 160 160
```

```r
form <- as.formula(paste(target, "~ ."))

model.wsrf.1 <- wsrf(form, data=train_raw.df, parallel=FALSE)
print(model.wsrf.1)
```

```
## A Weighted Subspace Random Forest model with 500 trees.
##
##   No. of variables tried at each split: 3
##         Minimum size of terminal nodes: 2
##                  Out-of-Bag Error Rate: 0.05
##                               Strength: 0.77
##                            Correlation: 0.07
##
## Confusion matrix:
##     1   0 class.error
## 1 125   6        0.05
## 0   7 118        0.06
```

```r
wdrf.fit <- predict(model.wsrf.1, newdata=test_raw.df, type="class")$class

##create confusion matrix
confusionMatrix(wdrf.fit, test_raw.df$Veracity, dnn = c("Prediction", "Reference"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1 29  3
##          0  0 32
##
##                Accuracy : 0.9531
##                  95% CI : (0.8691, 0.9902)
##     No Information Rate : 0.5469
##     P-Value [Acc > NIR] : 4.219e-13
##
##                   Kappa : 0.9062
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9143
##          Pos Pred Value : 0.9062
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4531
##          Detection Rate : 0.4531
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9571
##
##        'Positive' Class : 1
##
```

GBM

```
#install.packages("gbm")
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.2.2
```

```
## Loaded gbm 2.1.8.1
```

```
fit.gbm <- gbm(Veracity~. , data= train_raw.df,
               distribution = 'multinomial',
               cv.folds = 10,
               shrinkage = .01,
               n.minobsinnode = 10,
               n.trees = 200)
```

```
## Warning: Setting `distribution = "multinomial"` is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
pred <- predict.gbm(object = fit.gbm,
                   newdata = test_raw.df,
                   n.trees = 200,
                   type = "response")

##create confusion matrix
pred.labels = colnames(pred)[apply(pred, 1, which.max)]
result = data.frame(test_raw.df$Veracity, pred.labels)
caret::confusionMatrix(test_raw.df$Veracity, as.factor(pred.labels))
```

```
## Warning in confusionMatrix.default(test_raw.df$Veracity,
## as.factor(pred.labels)): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 19 16
##          1  7 22
##
##                Accuracy : 0.6406
##                  95% CI : (0.511, 0.7568)
##     No Information Rate : 0.5938
##     P-Value [Acc > NIR] : 0.26403
##
##                   Kappa : 0.2937
##
##  Mcnemar's Test P-Value : 0.09529
##
```

```
##             Sensitivity : 0.7308
##             Specificity : 0.5789
##          Pos Pred Value : 0.5429
##          Neg Pred Value : 0.7586
##              Prevalence : 0.4062
##          Detection Rate : 0.2969
##    Detection Prevalence : 0.5469
##       Balanced Accuracy : 0.6549
##
##        'Positive' Class : 0
##
```

Ensemble Learning

```
library(caretEnsemble)
```

```
## Warning: package 'caretEnsemble' was built under R version 4.2.2
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
set.seed(100)

control_stacking <- caret::trainControl(method="repeatedcv", number=5, repeats=2, savePredictions=TRUE,
algorithms_to_use <- c( 'glm', 'knn', 'svmPoly','svmLinear', 'wsrf',  'gbm')
stacked_models <- caretList(make.names(Veracity) ~., data=MU3D_Video_Level_Data.scaled, trControl=contro
```

```
## Warning in trControlCheck(x = trControl, y = target): x$savePredictions == TRUE
## is depreciated. Setting to 'final' instead.
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3583             nan     0.1000    0.0133
##      2        1.3382             nan     0.1000    0.0078
##      3        1.3199             nan     0.1000    0.0074
##      4        1.2949             nan     0.1000    0.0115
##      5        1.2803             nan     0.1000    0.0065
##      6        1.2638             nan     0.1000    0.0080
##      7        1.2515             nan     0.1000    0.0048
##      8        1.2403             nan     0.1000    0.0020
##      9        1.2342             nan     0.1000    0.0000
##     10        1.2277             nan     0.1000    0.0008
##     20        1.1541             nan     0.1000   -0.0007
```

```
##     40        1.0918             nan     0.1000   -0.0013
##     60        1.0528             nan     0.1000   -0.0017
##     80        1.0171             nan     0.1000   -0.0025
##    100        0.9827             nan     0.1000   -0.0034
##    120        0.9603             nan     0.1000   -0.0005
##    140        0.9401             nan     0.1000   -0.0003
##    150        0.9265             nan     0.1000   -0.0029
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2937             nan     0.1000    0.0447
##      2        1.2243             nan     0.1000    0.0341
##      3        1.1633             nan     0.1000    0.0268
##      4        1.1163             nan     0.1000    0.0191
##      5        1.0525             nan     0.1000    0.0248
##      6        1.0281             nan     0.1000    0.0088
##      7        1.0090             nan     0.1000    0.0054
##      8        0.9547             nan     0.1000    0.0265
##      9        0.9201             nan     0.1000    0.0133
##     10        0.8915             nan     0.1000    0.0108
##     20        0.8319             nan     0.1000   -0.0020
##     40        0.6226             nan     0.1000   -0.0008
##     60        0.4704             nan     0.1000    0.0066
##     80        0.3438             nan     0.1000    0.0006
##    100        0.2657             nan     0.1000    0.0056
##    120        0.2219             nan     0.1000   -0.0007
##    140        0.1809             nan     0.1000   -0.0012
##    150        0.1484             nan     0.1000   -0.0007
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2769             nan     0.1000    0.0499
##      2        1.1186             nan     0.1000    0.0781
##      3        1.0262             nan     0.1000    0.0450
##      4        0.9424             nan     0.1000    0.0384
##      5        0.9132             nan     0.1000    0.0090
##      6        0.8140             nan     0.1000    0.0500
##      7        0.7505             nan     0.1000    0.0312
##      8        0.6727             nan     0.1000    0.0387
##      9        0.6577             nan     0.1000    0.0059
##     10        0.6146             nan     0.1000    0.0198
##     20        0.3466             nan     0.1000    0.0153
##     40        0.2001             nan     0.1000   -0.0009
##     60        0.1530             nan     0.1000   -0.0011
##     80        0.1145             nan     0.1000   -0.0008
##    100        0.0866             nan     0.1000   -0.0004
##    120        0.0640             nan     0.1000    0.0003
##    140        0.0498             nan     0.1000   -0.0006
##    150        0.0427             nan     0.1000   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3562             nan     0.1000    0.0160
##      2        1.3223             nan     0.1000    0.0134
##      3        1.3044             nan     0.1000    0.0067
##      4        1.2859             nan     0.1000    0.0090
##      5        1.2711             nan     0.1000    0.0072
```

```
##      6        1.2507             nan     0.1000    0.0076
##      7        1.2345             nan     0.1000    0.0052
##      8        1.2265             nan     0.1000    0.0018
##      9        1.2143             nan     0.1000    0.0047
##     10        1.2049             nan     0.1000    0.0032
##     20        1.1338             nan     0.1000    0.0015
##     40        1.0737             nan     0.1000   -0.0015
##     60        1.0487             nan     0.1000   -0.0040
##     80        1.0207             nan     0.1000   -0.0019
##    100        0.9995             nan     0.1000   -0.0027
##    120        0.9691             nan     0.1000   -0.0012
##    140        0.9453             nan     0.1000   -0.0031
##    150        0.9358             nan     0.1000   -0.0040
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2910             nan     0.1000    0.0431
##      2        1.2202             nan     0.1000    0.0330
##      3        1.1317             nan     0.1000    0.0425
##      4        1.0814             nan     0.1000    0.0226
##      5        0.9961             nan     0.1000    0.0384
##      6        0.9359             nan     0.1000    0.0270
##      7        0.8857             nan     0.1000    0.0238
##      8        0.8759             nan     0.1000    0.0028
##      9        0.8480             nan     0.1000    0.0111
##     10        0.7955             nan     0.1000    0.0239
##     20        0.5915             nan     0.1000   -0.0015
##     40        0.4659             nan     0.1000   -0.0015
##     60        0.3820             nan     0.1000   -0.0015
##     80        0.2680             nan     0.1000    0.0038
##    100        0.1881             nan     0.1000   -0.0008
##    120        0.1379             nan     0.1000    0.0006
##    140        0.1184             nan     0.1000   -0.0004
##    150        0.1120             nan     0.1000   -0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2731             nan     0.1000    0.0528
##      2        1.1796             nan     0.1000    0.0390
##      3        1.0329             nan     0.1000    0.0719
##      4        0.9220             nan     0.1000    0.0520
##      5        0.8683             nan     0.1000    0.0213
##      6        0.7985             nan     0.1000    0.0302
##      7        0.7104             nan     0.1000    0.0424
##      8        0.6358             nan     0.1000    0.0373
##      9        0.5989             nan     0.1000    0.0147
##     10        0.5428             nan     0.1000    0.0271
##     20        0.2595             nan     0.1000    0.0104
##     40        0.1033             nan     0.1000    0.0001
##     60        0.0669             nan     0.1000    0.0006
##     80        0.0438             nan     0.1000    0.0005
##    100        0.0322             nan     0.1000   -0.0002
##    120        0.0261             nan     0.1000   -0.0001
##    140        0.0194             nan     0.1000   -0.0001
##    150        0.0165             nan     0.1000   -0.0001
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3539             nan     0.1000    0.0154
##      2        1.3272             nan     0.1000    0.0115
##      3        1.3051             nan     0.1000    0.0097
##      4        1.2835             nan     0.1000    0.0088
##      5        1.2661             nan     0.1000    0.0071
##      6        1.2490             nan     0.1000    0.0059
##      7        1.2337             nan     0.1000    0.0044
##      8        1.2229             nan     0.1000    0.0013
##      9        1.2141             nan     0.1000   -0.0006
##     10        1.2030             nan     0.1000    0.0022
##     20        1.1416             nan     0.1000    0.0029
##     40        1.0702             nan     0.1000   -0.0015
##     60        1.0323             nan     0.1000   -0.0004
##     80        1.0069             nan     0.1000   -0.0053
##    100        0.9787             nan     0.1000   -0.0023
##    120        0.9597             nan     0.1000   -0.0026
##    140        0.9397             nan     0.1000   -0.0023
##    150        0.9311             nan     0.1000   -0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2951             nan     0.1000    0.0406
##      2        1.2121             nan     0.1000    0.0339
##      3        1.1778             nan     0.1000    0.0131
##      4        1.0954             nan     0.1000    0.0386
##      5        1.0538             nan     0.1000    0.0161
##      6        0.9830             nan     0.1000    0.0338
##      7        0.9289             nan     0.1000    0.0223
##      8        0.8704             nan     0.1000    0.0283
##      9        0.8325             nan     0.1000    0.0180
##     10        0.7839             nan     0.1000    0.0236
##     20        0.6163             nan     0.1000   -0.0021
##     40        0.4384             nan     0.1000   -0.0001
##     60        0.3764             nan     0.1000   -0.0018
##     80        0.2848             nan     0.1000   -0.0009
##    100        0.2170             nan     0.1000   -0.0016
##    120        0.1841             nan     0.1000   -0.0004
##    140        0.1608             nan     0.1000   -0.0005
##    150        0.1460             nan     0.1000   -0.0009
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2663             nan     0.1000    0.0547
##      2        1.1160             nan     0.1000    0.0761
##      3        1.0325             nan     0.1000    0.0399
##      4        0.9214             nan     0.1000    0.0551
##      5        0.8226             nan     0.1000    0.0494
##      6        0.7612             nan     0.1000    0.0278
##      7        0.7159             nan     0.1000    0.0212
##      8        0.6507             nan     0.1000    0.0314
##      9        0.6098             nan     0.1000    0.0185
##     10        0.5527             nan     0.1000    0.0279
##     20        0.2811             nan     0.1000    0.0035
##     40        0.1553             nan     0.1000   -0.0007
##     60        0.1123             nan     0.1000   -0.0009
```

```
##     80        0.0891             nan     0.1000   -0.0009
##    100        0.0698             nan     0.1000   -0.0007
##    120        0.0595             nan     0.1000   -0.0003
##    140        0.0443             nan     0.1000    0.0009
##    150        0.0384             nan     0.1000   -0.0003
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3519             nan     0.1000    0.0167
##      2        1.3199             nan     0.1000    0.0136
##      3        1.2876             nan     0.1000    0.0131
##      4        1.2569             nan     0.1000    0.0082
##      5        1.2359             nan     0.1000    0.0083
##      6        1.2204             nan     0.1000    0.0058
##      7        1.2083             nan     0.1000    0.0066
##      8        1.1980             nan     0.1000    0.0033
##      9        1.1847             nan     0.1000    0.0038
##     10        1.1766             nan     0.1000    0.0025
##     20        1.1266             nan     0.1000   -0.0070
##     40        1.0782             nan     0.1000   -0.0053
##     60        1.0272             nan     0.1000   -0.0012
##     80        1.0032             nan     0.1000   -0.0026
##    100        0.9713             nan     0.1000   -0.0044
##    120        0.9513             nan     0.1000   -0.0022
##    140        0.9266             nan     0.1000   -0.0020
##    150        0.9199             nan     0.1000   -0.0012
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2985             nan     0.1000    0.0440
##      2        1.2191             nan     0.1000    0.0368
##      3        1.1532             nan     0.1000    0.0313
##      4        1.0831             nan     0.1000    0.0316
##      5        1.0175             nan     0.1000    0.0286
##      6        0.9506             nan     0.1000    0.0333
##      7        0.9005             nan     0.1000    0.0238
##      8        0.8686             nan     0.1000    0.0129
##      9        0.8166             nan     0.1000    0.0248
##     10        0.7902             nan     0.1000    0.0104
##     20        0.5823             nan     0.1000    0.0007
##     40        0.4651             nan     0.1000   -0.0009
##     60        0.4138             nan     0.1000   -0.0007
##     80        0.3336             nan     0.1000    0.0044
##    100        0.2688             nan     0.1000   -0.0020
##    120        0.2221             nan     0.1000   -0.0010
##    140        0.1734             nan     0.1000   -0.0004
##    150        0.1640             nan     0.1000   -0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2105             nan     0.1000    0.0892
##      2        1.0937             nan     0.1000    0.0554
##      3        0.9982             nan     0.1000    0.0455
##      4        0.9201             nan     0.1000    0.0348
##      5        0.8556             nan     0.1000    0.0310
##      6        0.7615             nan     0.1000    0.0445
##      7        0.7230             nan     0.1000    0.0181
```

```
##      8        0.6524             nan     0.1000    0.0358
##      9        0.6075             nan     0.1000    0.0214
##     10        0.5494             nan     0.1000    0.0280
##     20        0.2620             nan     0.1000    0.0087
##     40        0.1381             nan     0.1000    0.0000
##     60        0.1029             nan     0.1000   -0.0004
##     80        0.0834             nan     0.1000   -0.0002
##    100        0.0629             nan     0.1000   -0.0006
##    120        0.0487             nan     0.1000   -0.0002
##    140        0.0396             nan     0.1000   -0.0002
##    150        0.0339             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3533             nan     0.1000    0.0181
##      2        1.3196             nan     0.1000    0.0145
##      3        1.2960             nan     0.1000    0.0111
##      4        1.2695             nan     0.1000    0.0111
##      5        1.2514             nan     0.1000    0.0052
##      6        1.2395             nan     0.1000    0.0044
##      7        1.2220             nan     0.1000    0.0065
##      8        1.2106             nan     0.1000    0.0048
##      9        1.1974             nan     0.1000    0.0033
##     10        1.1862             nan     0.1000    0.0031
##     20        1.1321             nan     0.1000   -0.0031
##     40        1.0673             nan     0.1000   -0.0022
##     60        1.0225             nan     0.1000   -0.0030
##     80        0.9930             nan     0.1000   -0.0064
##    100        0.9647             nan     0.1000   -0.0009
##    120        0.9449             nan     0.1000   -0.0018
##    140        0.9193             nan     0.1000   -0.0027
##    150        0.9068             nan     0.1000   -0.0013
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2663             nan     0.1000    0.0547
##      2        1.1833             nan     0.1000    0.0408
##      3        1.1114             nan     0.1000    0.0367
##      4        1.0585             nan     0.1000    0.0229
##      5        0.9853             nan     0.1000    0.0383
##      6        0.9274             nan     0.1000    0.0285
##      7        0.9037             nan     0.1000    0.0104
##      8        0.8494             nan     0.1000    0.0241
##      9        0.7968             nan     0.1000    0.0248
##     10        0.7710             nan     0.1000    0.0083
##     20        0.5931             nan     0.1000    0.0055
##     40        0.4628             nan     0.1000    0.0006
##     60        0.3454             nan     0.1000    0.0004
##     80        0.2582             nan     0.1000   -0.0005
##    100        0.1920             nan     0.1000   -0.0006
##    120        0.1675             nan     0.1000   -0.0010
##    140        0.1295             nan     0.1000    0.0007
##    150        0.1189             nan     0.1000    0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2165             nan     0.1000    0.0833
```

```
##      2        1.1405             nan     0.1000    0.0342
##      3        1.0471             nan     0.1000    0.0415
##      4        0.9261             nan     0.1000    0.0592
##      5        0.8543             nan     0.1000    0.0304
##      6        0.7629             nan     0.1000    0.0449
##      7        0.6850             nan     0.1000    0.0393
##      8        0.6170             nan     0.1000    0.0332
##      9        0.5570             nan     0.1000    0.0285
##     10        0.5063             nan     0.1000    0.0247
##     20        0.2631             nan     0.1000    0.0096
##     40        0.1302             nan     0.1000   -0.0005
##     60        0.0995             nan     0.1000    0.0003
##     80        0.0703             nan     0.1000   -0.0005
##    100        0.0529             nan     0.1000   -0.0008
##    120        0.0404             nan     0.1000   -0.0002
##    140        0.0314             nan     0.1000    0.0002
##    150        0.0293             nan     0.1000   -0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3554             nan     0.1000    0.0161
##      2        1.3251             nan     0.1000    0.0116
##      3        1.3094             nan     0.1000    0.0047
##      4        1.2825             nan     0.1000    0.0033
##      5        1.2652             nan     0.1000    0.0070
##      6        1.2549             nan     0.1000    0.0015
##      7        1.2338             nan     0.1000    0.0082
##      8        1.2217             nan     0.1000    0.0067
##      9        1.2157             nan     0.1000   -0.0003
##     10        1.2039             nan     0.1000    0.0024
##     20        1.1335             nan     0.1000    0.0014
##     40        1.0678             nan     0.1000   -0.0011
##     60        1.0319             nan     0.1000   -0.0002
##     80        0.9962             nan     0.1000   -0.0034
##    100        0.9724             nan     0.1000   -0.0012
##    120        0.9456             nan     0.1000   -0.0045
##    140        0.9231             nan     0.1000   -0.0008
##    150        0.9179             nan     0.1000   -0.0026
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2862             nan     0.1000    0.0480
##      2        1.2128             nan     0.1000    0.0334
##      3        1.1442             nan     0.1000    0.0327
##      4        1.0929             nan     0.1000    0.0247
##      5        1.0390             nan     0.1000    0.0289
##      6        1.0000             nan     0.1000    0.0188
##      7        0.9630             nan     0.1000    0.0168
##      8        0.9177             nan     0.1000    0.0192
##      9        0.8574             nan     0.1000    0.0289
##     10        0.8144             nan     0.1000    0.0197
##     20        0.5949             nan     0.1000   -0.0005
##     40        0.4647             nan     0.1000    0.0084
##     60        0.3544             nan     0.1000   -0.0007
##     80        0.2892             nan     0.1000    0.0001
##    100        0.2296             nan     0.1000    0.0005
```

```
##    120        0.1639             nan     0.1000   -0.0017
##    140        0.1344             nan     0.1000   -0.0007
##    150        0.1224             nan     0.1000   -0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2835             nan     0.1000    0.0474
##      2        1.1702             nan     0.1000    0.0514
##      3        1.0360             nan     0.1000    0.0657
##      4        0.9537             nan     0.1000    0.0377
##      5        0.8822             nan     0.1000    0.0344
##      6        0.8260             nan     0.1000    0.0247
##      7        0.8054             nan     0.1000    0.0043
##      8        0.7506             nan     0.1000    0.0241
##      9        0.6953             nan     0.1000    0.0287
##     10        0.6232             nan     0.1000    0.0355
##     20        0.2917             nan     0.1000    0.0063
##     40        0.1268             nan     0.1000   -0.0011
##     60        0.0898             nan     0.1000   -0.0007
##     80        0.0735             nan     0.1000   -0.0007
##    100        0.0577             nan     0.1000   -0.0003
##    120        0.0435             nan     0.1000    0.0014
##    140        0.0342             nan     0.1000   -0.0001
##    150        0.0323             nan     0.1000   -0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3515             nan     0.1000    0.0143
##      2        1.3225             nan     0.1000    0.0145
##      3        1.3060             nan     0.1000    0.0110
##      4        1.2858             nan     0.1000    0.0053
##      5        1.2649             nan     0.1000    0.0044
##      6        1.2506             nan     0.1000    0.0060
##      7        1.2338             nan     0.1000    0.0058
##      8        1.2236             nan     0.1000    0.0037
##      9        1.2130             nan     0.1000    0.0049
##     10        1.2088             nan     0.1000   -0.0035
##     20        1.1515             nan     0.1000   -0.0029
##     40        1.0930             nan     0.1000   -0.0025
##     60        1.0591             nan     0.1000   -0.0026
##     80        1.0306             nan     0.1000   -0.0017
##    100        1.0109             nan     0.1000   -0.0021
##    120        0.9929             nan     0.1000   -0.0043
##    140        0.9641             nan     0.1000   -0.0012
##    150        0.9541             nan     0.1000   -0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2783             nan     0.1000    0.0506
##      2        1.1842             nan     0.1000    0.0446
##      3        1.1201             nan     0.1000    0.0235
##      4        1.0644             nan     0.1000    0.0242
##      5        0.9842             nan     0.1000    0.0412
##      6        0.9231             nan     0.1000    0.0282
##      7        0.9061             nan     0.1000    0.0050
##      8        0.8839             nan     0.1000    0.0115
##      9        0.8320             nan     0.1000    0.0222
```

```
##     10        0.8203             nan     0.1000    0.0045
##     20        0.6347             nan     0.1000    0.0223
##     40        0.5379             nan     0.1000   -0.0015
##     60        0.4212             nan     0.1000   -0.0008
##     80        0.3151             nan     0.1000    0.0001
##    100        0.2510             nan     0.1000   -0.0012
##    120        0.2000             nan     0.1000    0.0013
##    140        0.1468             nan     0.1000    0.0021
##    150        0.1223             nan     0.1000   -0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3321             nan     0.1000    0.0151
##      2        1.1716             nan     0.1000    0.0777
##      3        1.0262             nan     0.1000    0.0722
##      4        0.9617             nan     0.1000    0.0245
##      5        0.8523             nan     0.1000    0.0554
##      6        0.7961             nan     0.1000    0.0309
##      7        0.7104             nan     0.1000    0.0430
##      8        0.6331             nan     0.1000    0.0370
##      9        0.5697             nan     0.1000    0.0321
##     10        0.5218             nan     0.1000    0.0229
##     20        0.2631             nan     0.1000    0.0031
##     40        0.0963             nan     0.1000   -0.0003
##     60        0.0623             nan     0.1000    0.0004
##     80        0.0397             nan     0.1000   -0.0001
##    100        0.0287             nan     0.1000    0.0001
##    120        0.0219             nan     0.1000   -0.0003
##    140        0.0176             nan     0.1000    0.0003
##    150        0.0158             nan     0.1000   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3510             nan     0.1000    0.0164
##      2        1.3199             nan     0.1000    0.0155
##      3        1.3048             nan     0.1000    0.0044
##      4        1.2789             nan     0.1000    0.0111
##      5        1.2602             nan     0.1000    0.0084
##      6        1.2452             nan     0.1000    0.0082
##      7        1.2326             nan     0.1000    0.0053
##      8        1.2158             nan     0.1000    0.0052
##      9        1.2114             nan     0.1000   -0.0006
##     10        1.2016             nan     0.1000    0.0038
##     20        1.1399             nan     0.1000   -0.0034
##     40        1.0943             nan     0.1000   -0.0038
##     60        1.0531             nan     0.1000   -0.0018
##     80        1.0243             nan     0.1000   -0.0026
##    100        1.0018             nan     0.1000   -0.0036
##    120        0.9730             nan     0.1000   -0.0008
##    140        0.9511             nan     0.1000   -0.0016
##    150        0.9423             nan     0.1000   -0.0051
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2926             nan     0.1000    0.0406
##      2        1.1940             nan     0.1000    0.0411
##      3        1.1254             nan     0.1000    0.0322
```

```
##      4        1.0547             nan     0.1000    0.0315
##      5        0.9893             nan     0.1000    0.0299
##      6        0.9453             nan     0.1000    0.0204
##      7        0.8863             nan     0.1000    0.0278
##      8        0.8487             nan     0.1000    0.0189
##      9        0.8064             nan     0.1000    0.0198
##     10        0.7643             nan     0.1000    0.0195
##     20        0.5916             nan     0.1000    0.0097
##     40        0.4255             nan     0.1000    0.0029
##     60        0.3512             nan     0.1000   -0.0011
##     80        0.3044             nan     0.1000   -0.0010
##    100        0.2415             nan     0.1000   -0.0005
##    120        0.2136             nan     0.1000   -0.0016
##    140        0.1718             nan     0.1000   -0.0002
##    150        0.1588             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2078             nan     0.1000    0.0864
##      2        1.0797             nan     0.1000    0.0624
##      3        0.9563             nan     0.1000    0.0605
##      4        0.8547             nan     0.1000    0.0504
##      5        0.7947             nan     0.1000    0.0304
##      6        0.7243             nan     0.1000    0.0354
##      7        0.6772             nan     0.1000    0.0209
##      8        0.6293             nan     0.1000    0.0240
##      9        0.5716             nan     0.1000    0.0277
##     10        0.5182             nan     0.1000    0.0244
##     20        0.2589             nan     0.1000    0.0087
##     40        0.1297             nan     0.1000    0.0002
##     60        0.0956             nan     0.1000   -0.0004
##     80        0.0765             nan     0.1000   -0.0004
##    100        0.0640             nan     0.1000   -0.0003
##    120        0.0501             nan     0.1000   -0.0004
##    140        0.0419             nan     0.1000   -0.0008
##    150        0.0373             nan     0.1000   -0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3475             nan     0.1000    0.0179
##      2        1.3175             nan     0.1000    0.0128
##      3        1.2842             nan     0.1000    0.0124
##      4        1.2637             nan     0.1000    0.0108
##      5        1.2463             nan     0.1000    0.0075
##      6        1.2243             nan     0.1000    0.0057
##      7        1.2111             nan     0.1000    0.0060
##      8        1.1992             nan     0.1000    0.0062
##      9        1.1875             nan     0.1000    0.0036
##     10        1.1781             nan     0.1000    0.0021
##     20        1.1086             nan     0.1000   -0.0018
##     40        1.0456             nan     0.1000   -0.0013
##     60        1.0124             nan     0.1000   -0.0024
##     80        0.9821             nan     0.1000   -0.0036
##    100        0.9524             nan     0.1000   -0.0020
##    120        0.9287             nan     0.1000   -0.0044
##    140        0.9075             nan     0.1000   -0.0034
```

```
##    150        0.9045             nan     0.1000   -0.0003
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2775             nan     0.1000    0.0583
##      2        1.1865             nan     0.1000    0.0385
##      3        1.1085             nan     0.1000    0.0370
##      4        1.0617             nan     0.1000    0.0200
##      5        1.0070             nan     0.1000    0.0258
##      6        0.9678             nan     0.1000    0.0138
##      7        0.9335             nan     0.1000    0.0160
##      8        0.8985             nan     0.1000    0.0145
##      9        0.8576             nan     0.1000    0.0152
##     10        0.8420             nan     0.1000    0.0049
##     20        0.6234             nan     0.1000   -0.0001
##     40        0.4655             nan     0.1000    0.0021
##     60        0.3536             nan     0.1000    0.0007
##     80        0.2844             nan     0.1000   -0.0012
##    100        0.2451             nan     0.1000   -0.0003
##    120        0.2098             nan     0.1000   -0.0010
##    140        0.1717             nan     0.1000   -0.0007
##    150        0.1674             nan     0.1000   -0.0009
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2646             nan     0.1000    0.0614
##      2        1.2187             nan     0.1000    0.0174
##      3        1.0767             nan     0.1000    0.0728
##      4        1.0486             nan     0.1000    0.0122
##      5        0.9441             nan     0.1000    0.0509
##      6        0.8712             nan     0.1000    0.0343
##      7        0.8386             nan     0.1000    0.0144
##      8        0.7800             nan     0.1000    0.0257
##      9        0.7374             nan     0.1000    0.0186
##     10        0.6923             nan     0.1000    0.0204
##     20        0.3854             nan     0.1000    0.0176
##     40        0.1881             nan     0.1000    0.0059
##     60        0.1368             nan     0.1000    0.0011
##     80        0.1115             nan     0.1000   -0.0002
##    100        0.0886             nan     0.1000   -0.0010
##    120        0.0703             nan     0.1000   -0.0002
##    140        0.0524             nan     0.1000   -0.0001
##    150        0.0473             nan     0.1000   -0.0006
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3540             nan     0.1000    0.0108
##      2        1.3228             nan     0.1000    0.0121
##      3        1.2968             nan     0.1000    0.0111
##      4        1.2808             nan     0.1000    0.0096
##      5        1.2635             nan     0.1000    0.0070
##      6        1.2617             nan     0.1000   -0.0023
##      7        1.2472             nan     0.1000    0.0052
##      8        1.2317             nan     0.1000    0.0072
##      9        1.2241             nan     0.1000   -0.0008
##     10        1.2115             nan     0.1000    0.0025
##     20        1.1355             nan     0.1000   -0.0017
```

```
##     40        1.0758             nan     0.1000   -0.0034
##     60        1.0409             nan     0.1000   -0.0043
##     80        1.0110             nan     0.1000   -0.0020
##    100        0.9871             nan     0.1000   -0.0008
##    120        0.9617             nan     0.1000   -0.0051
##    140        0.9384             nan     0.1000   -0.0019
##    150        0.9329             nan     0.1000   -0.0050
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2972             nan     0.1000    0.0405
##      2        1.2183             nan     0.1000    0.0378
##      3        1.1504             nan     0.1000    0.0340
##      4        1.0883             nan     0.1000    0.0246
##      5        1.0674             nan     0.1000    0.0096
##      6        1.0445             nan     0.1000    0.0076
##      7        0.9990             nan     0.1000    0.0199
##      8        0.9224             nan     0.1000    0.0338
##      9        0.9044             nan     0.1000    0.0045
##     10        0.8782             nan     0.1000    0.0088
##     20        0.6760             nan     0.1000   -0.0009
##     40        0.5285             nan     0.1000    0.0001
##     60        0.4527             nan     0.1000   -0.0016
##     80        0.3832             nan     0.1000   -0.0011
##    100        0.3165             nan     0.1000   -0.0008
##    120        0.2496             nan     0.1000    0.0040
##    140        0.2136             nan     0.1000    0.0003
##    150        0.1925             nan     0.1000   -0.0007
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2326             nan     0.1000    0.0748
##      2        1.1773             nan     0.1000    0.0265
##      3        1.0755             nan     0.1000    0.0483
##      4        1.0347             nan     0.1000    0.0162
##      5        0.9491             nan     0.1000    0.0416
##      6        0.9037             nan     0.1000    0.0191
##      7        0.8115             nan     0.1000    0.0442
##      8        0.7350             nan     0.1000    0.0377
##      9        0.7111             nan     0.1000    0.0094
##     10        0.6389             nan     0.1000    0.0368
##     20        0.4444             nan     0.1000    0.0010
##     40        0.2072             nan     0.1000   -0.0002
##     60        0.1496             nan     0.1000   -0.0007
##     80        0.1227             nan     0.1000   -0.0010
##    100        0.1009             nan     0.1000    0.0004
##    120        0.0758             nan     0.1000   -0.0005
##    140        0.0601             nan     0.1000   -0.0002
##    150        0.0541             nan     0.1000   -0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3345             nan     0.1000    0.0165
##      2        1.2072             nan     0.1000    0.0638
##      3        1.1068             nan     0.1000    0.0478
##      4        0.9793             nan     0.1000    0.0644
##      5        0.9074             nan     0.1000    0.0343
```

```
##      6        0.8415             nan     0.1000    0.0262
##      7        0.7672             nan     0.1000    0.0359
##      8        0.6904             nan     0.1000    0.0393
##      9        0.6456             nan     0.1000    0.0225
##     10        0.5947             nan     0.1000    0.0253
##     20        0.3133             nan     0.1000    0.0073
##     40        0.1374             nan     0.1000    0.0003
##     50        0.1235             nan     0.1000   -0.0015
```

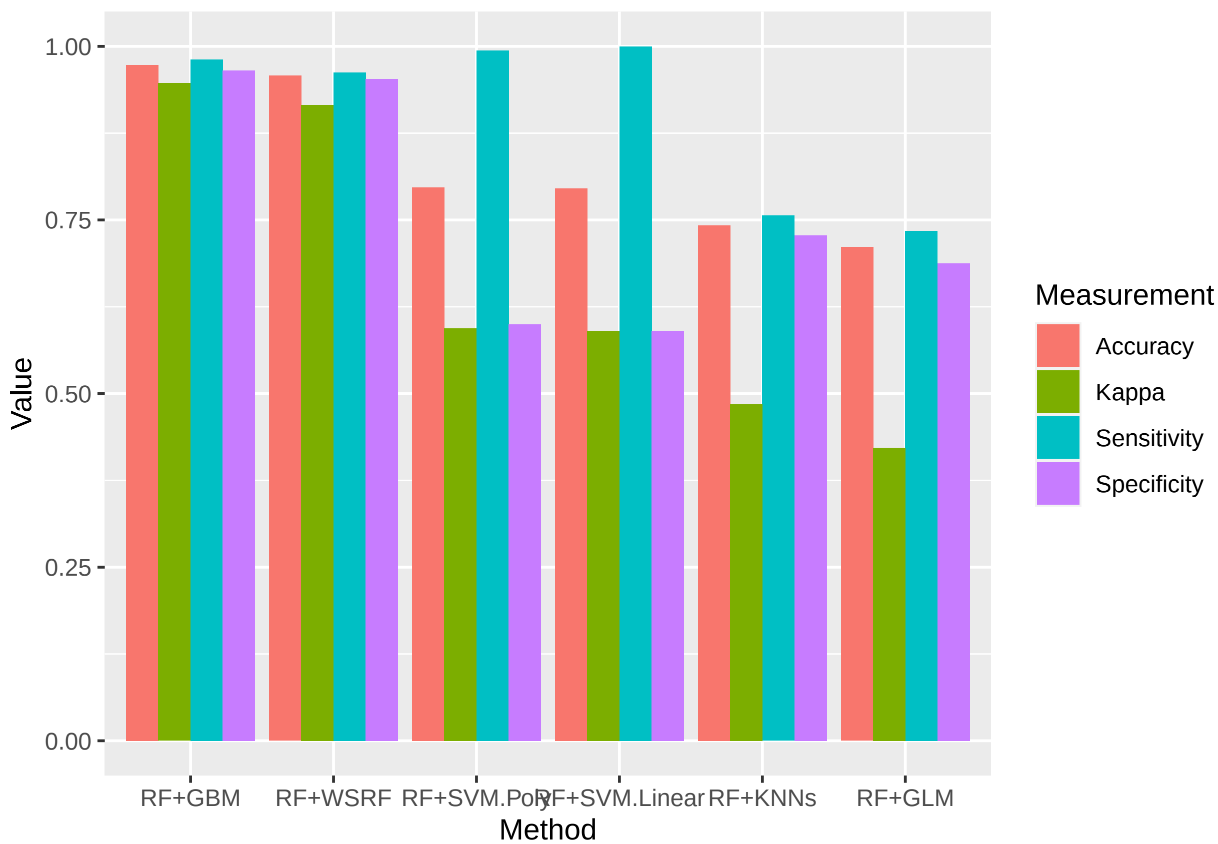
```
stacking_results <- resamples(stacked_models)

stacking_summary<- summary(stacking_results)
stacking_summary
```

```
##
## Call:
## summary.resamples(object = stacking_results)
##
## Models: glm, knn, svmPoly, svmLinear, wsrf, gbm
## Number of resamples: 10
##
## Accuracy
##               Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## glm       0.609375 0.6757812 0.7031250 0.7093750 0.7500000 0.812500    0
## knn       0.640625 0.6914062 0.7421875 0.7375000 0.7656250 0.828125    0
## svmPoly   0.765625 0.7851562 0.8125000 0.8031250 0.8125000 0.843750    0
## svmLinear 0.765625 0.7851562 0.7968750 0.8015625 0.8125000 0.843750    0
## wsrf      0.906250 0.9375000 0.9453125 0.9531250 0.9648438 1.000000    0
## gbm       0.953125 0.9570312 0.9843750 0.9765625 0.9843750 1.000000    0
##
## Kappa
##              Min.   1st Qu.   Median     Mean   3rd Qu.    Max. NA's
## glm       0.21875 0.3515625 0.406250 0.418750 0.5000000 0.62500    0
## knn       0.28125 0.3828125 0.484375 0.475000 0.5312500 0.65625    0
## svmPoly   0.53125 0.5703125 0.625000 0.606250 0.6250000 0.68750    0
## svmLinear 0.53125 0.5703125 0.593750 0.603125 0.6250000 0.68750    0
## wsrf      0.81250 0.8750000 0.890625 0.906250 0.9296875 1.00000    0
## gbm       0.90625 0.9140625 0.968750 0.953125 0.9687500 1.00000    0
```

Plot results

```
glm_cm<- confusionMatrix(stacked_models$glm$pred$pred, stacked_models$glm$pred$obs)
knn_cm<- confusionMatrix(stacked_models$knn$pred$pred, stacked_models$knn$pred$obs)
svmPoly_cm<- confusionMatrix(stacked_models$svmPoly$pred$pred, stacked_models$svmPoly$pred$obs)
svmLinear_cm<- confusionMatrix(stacked_models$svmLinear$pred$pred, stacked_models$svmLinear$pred$obs)
wsrf_cm<- confusionMatrix(stacked_models$wsrf$pred$pred, stacked_models$wsrf$pred$obs)
gbm_cm<- confusionMatrix(stacked_models$gbm$pred$pred, stacked_models$gbm$pred$obs)

data <- data.frame(matrix(c(0.7109, 0.7422, 0.7969, 0.7953, 0.9578, 0.9734,
                            0.7344, 0.7562, 0.9938, 1.0000, 0.9625, 0.9812,
                            0.6875, 0.7281, 0.6000, 0.5906, 0.9531, 0.9656,
```

```
                              0.4219, 0.4844, 0.5938, 0.5906, 0.9156, 0.9469),6,4))
row <- c("RF+GLM", "RF+KNNs", "RF+SVM.Poly", "RF+SVM.Linear", "RF+WSRF", "RF+GBM")
col <- c("Accuracy", "Sensitivity", "Specificity", "Kappa")
colnames(data) <- col
data$Method <- row
rownames(data) <- c(1:6)
data1<- data[,c(5,1,2,3,4)]

#write.csv(data, "ensemble_result.csv")
#plotting
library(ggplot2)

data2 <- tidyr::gather(data1, key="Measurement", value="Value", 2:5)

# Grouped
ggplot(data2, aes(fill=Measurement, y=Value, x=reorder(Method, -Value))) +
  geom_bar(position="dodge", stat="identity")+
  xlab("Method")
```



NLP in Transcription

document summarize

```r
# write summerizer function
library(textmineR)
```

```
## Warning: package 'textmineR' was built under R version 4.2.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.2.2

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'textmineR'

## The following object is masked from 'package:Matrix':
##
##     update

## The following object is masked from 'package:stats':
##
##     update
```

```r
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.2.2

##
## Attaching package: 'igraph'

## The following object is masked from 'package:wsrf':
##
##     strength

## The following object is masked from 'package:class':
##
##     knn

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
tcm <- CreateTcm(doc_vec = MU3D_Video_Level_Data$Transcription,
                 skipgram_window = 10,
                 verbose = FALSE,
                 cpus = 2)
```

```
## as(<dgTMatrix>, "dgCMatrix") is deprecated since Matrix 1.5-0; do as(., "CsparseMatrix") instead
```

```
# use LDA to get embeddings into probability space
# This will take considerably longer as the TCM matrix has many more rows
# than a DTM
embeddings <- FitLdaModel(dtm = tcm,
                          k = 50,
                          iterations = 200,
                          burnin = 180,
                          alpha = 0.1,
                          beta = 0.05,
                          optimize_alpha = TRUE,
                          calc_likelihood = FALSE,
                          calc_coherence = FALSE,
                          calc_r2 = FALSE,
                          cpus = 2)

summarizer <- function(doc, gamma) {

  # recursive fanciness to handle multiple docs at once
  if (length(doc) > 1 )
    # use a try statement to catch any weirdness that may arise
    return(sapply(doc, function(d) try(summarizer(d, gamma))))

  # parse it into sentences
  sent <- stringi::stri_split_boundaries(doc, type = "sentence")[[ 1 ]]
```

```
  names(sent) <- seq_along(sent) # so we know index and order

  # embed the sentences in the model
  e <- CreateDtm(sent, ngram_window = c(1,1), verbose = FALSE, cpus = 2)

  # remove any documents with 2 or fewer words
  e <- e[ rowSums(e) > 2 , ]

  vocab <- intersect(colnames(e), colnames(gamma))

  e <- e / rowSums(e)

  e <- e[ , vocab ] %*% t(gamma[ , vocab ])

  e <- as.matrix(e)

  # get the pairwise distances between each embedded sentence
  e_dist <- CalcHellingerDist(e)

  # turn into a similarity matrix
  g <- (1 - e_dist) * 100

  # we don't need sentences connected to themselves
  diag(g) <- 0

  # turn into a nearest-neighbor graph
  g <- apply(g, 1, function(x){
    x[ x < sort(x, decreasing = TRUE)[ 3 ] ] <- 0
    x
  })

  # by taking pointwise max, we'll make the matrix symmetric again
  g <- pmax(g, t(g))

  g <- graph.adjacency(g, mode = "undirected", weighted = TRUE)

  # calculate eigenvector centrality
  ev <- evcent(g)

  # format the result
  result <- sent[ names(ev$vector)[ order(ev$vector, decreasing = TRUE)[ 1:3 ] ] ]

  result <- result[ order(as.numeric(names(result))) ]

  paste(result, collapse = " ")
}
```

Summarize text

```
docs <- MU3D_Video_Level_Data$Transcription[1]
sums <- summarizer(docs, gamma = embeddings$gamma)
docs
```

```
## [1] "My best friend is a really nice person. Um. She’s always kind to everyone. She continues to jus
```

```
sums
```

```
## [1] "She has taught me so much throughout, like I’ve known her maybe a year and a half and she’s tau
```