# MU3D Lie Detection Report

Kun Bu

2022-07-29

```r
library(reshape2)
library(factoextra)
```

```
## Loading required package: ggplot2

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

```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.1.2

##
## Attaching package: 'psych'

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

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(FactoMineR)
library(devtools)
```

```
## Loading required package: usethis

## Warning: package 'usethis' was built under R version 4.1.2
```

```r
install_github('sinhrks/ggfortify')
```

```
## Skipping install of 'ggfortify' from a github remote, the SHA1 (8e3e7df6) has not changed since last
##   Use `force = TRUE` to force installation
```

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

```
## Warning: package 'e1071' was built under R version 4.1.2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: lattice
```

```r
#import video level dataset
MU3D_Video_Level_Data0 <- read.csv("MU3D_Video_Level_Data.csv")
head(MU3D_Video_Level_Data0)
```

```
##     VideoID Valence Veracity Sex Race VidLength_ms VidLength_sec WordCount
```

```
## 1 BF001_1PT       1        1   0    0        38783         38.78       110
## 2 BF001_2NL       0        0   0    0        37120         37.12        88
## 3 BF001_3NT       0        1   0    0        38484         38.48       120
## 4 BF001_4PL       1        0   0    0        38026         38.03       124
## 5 BF002_1PT       1        1   0    0        36351         36.35        91
## 6 BF002_2NL       0        0   0    0        36650         36.65        73
##   Accuracy TruthProp Attractive Trustworthy Anxious
## 1     0.77      0.77       4.55        4.32    3.18
## 2     0.40      0.60       3.55        3.75    3.05
## 3     0.77      0.77       3.27        3.95    2.82
## 4     0.58      0.42       4.05        4.05    3.11
## 5     0.59      0.59       4.86        4.36    3.32
## 6     0.33      0.67       5.05        4.62    2.33
##
## 1                                                                     My best friend
## 2
## 3                                      So this specific person is actually just a really mean and neg
## 4 This person is actually a really kind person. She has so many friends. She's very popular. Everyone
## 5
## 6
```

## Boxplot for variables

```r
#remove veractiy first
MU3D_Video_Level_Data <- MU3D_Video_Level_Data0[,-3]
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)
```

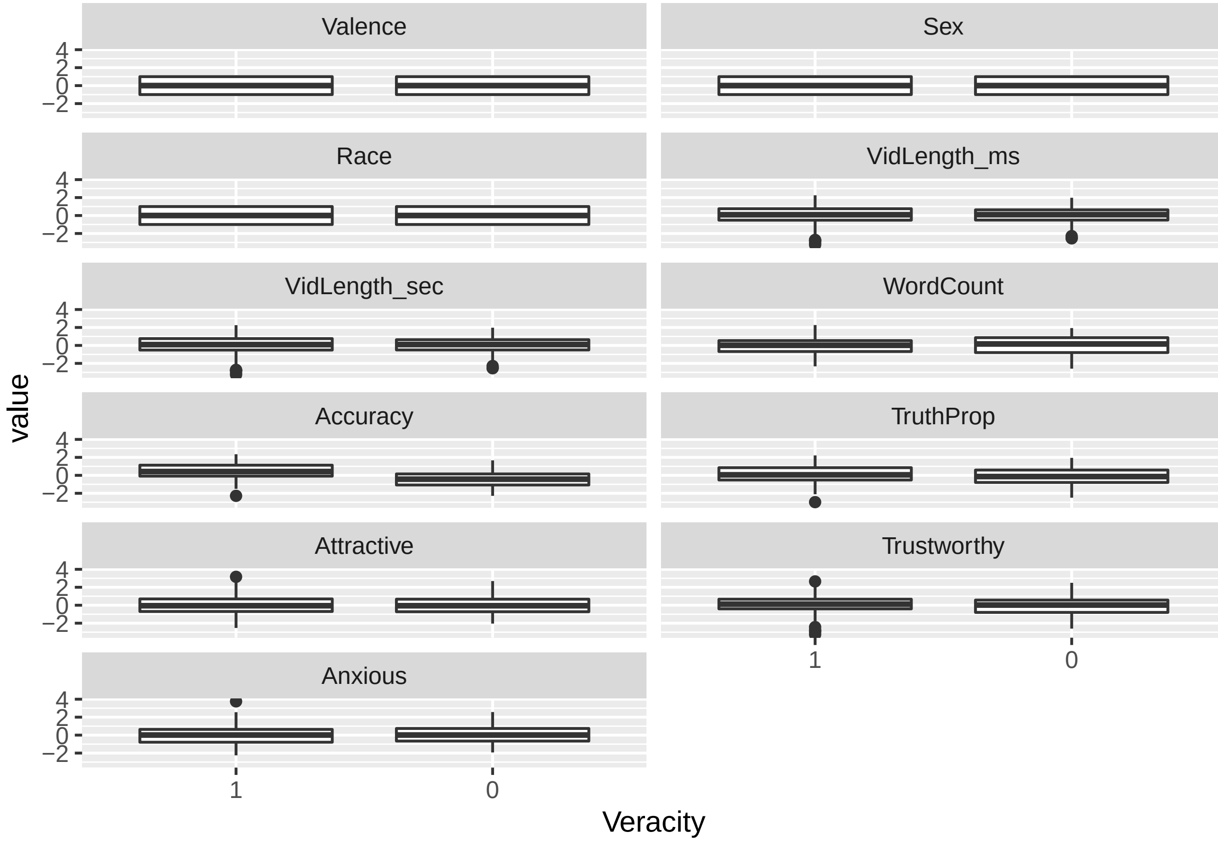
```
##  [1] "VideoID"       "Valence"       "Sex"           "Race"
##  [5] "VidLength_ms"  "VidLength_sec" "WordCount"     "Accuracy"
##  [9] "TruthProp"     "Attractive"    "Trustworthy"   "Anxious"
## [13] "Transcription"
```

```r
MU3D_Video_Level_Data.scaled <- data.frame(scale(MU3D_Video_Level_Data[,-c(1,13)]))

#level veractiy
levels0 <- unique(c(MU3D_Video_Level_Data0$Veracity, MU3D_Video_Level_Data0$Veracity))
```

```r
#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=2)
```




### SVM split train and test 80/20

```r
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, ])
test_raw.df <- as.data.frame(MU3D_Video_Level_Data.scaled[-train_ind_raw, ])
levels <- unique(c(train_raw.df$Veracity, test_raw.df$Veracity))
test_raw.df$Veracity  <- factor(test_raw.df$Veracity, levels=levels)
train_raw.df$Veracity <- factor(train_raw.df$Veracity, levels=levels)

# tuning best svm model for linear kernel
linear.tune <- tune.svm(Veracity ~ ., data = train_raw.df,
                        kernel = "linear",
                        cost = c(0.001, 0.01, 0.1, 1, 5, 10))
summary(linear.tune) #best cost is 1, misclassification rate no larger than 25%
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.2193846
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.5549231 0.03879886
## 2 1e-02 0.2624615 0.06741279
## 3 1e-01 0.2310769 0.07054517
## 4 1e+00 0.2233846 0.07465652
## 5 5e+00 0.2193846 0.07708086
## 6 1e+01 0.2193846 0.07708086
```

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

```
##
## linear.test  0  1
##           0 32 22
##           1  0 10
```

```r
confusionMatrix(linear.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference")) # 76.6% accuracy
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32 22
##          1  0 10
##
##                Accuracy : 0.6562
##                  95% CI : (0.527, 0.7705)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.008429
##
##                   Kappa : 0.3125
##
##  Mcnemar's Test P-Value : 7.562e-06
##
##             Sensitivity : 1.0000
##             Specificity : 0.3125
##          Pos Pred Value : 0.5926
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.8438
##       Balanced Accuracy : 0.6562
```

```
##
##         'Positive' Class : 0
##
```

```r
#
# # tuning best svm model for sigmoid kernel
# sigmoid.tune <- tune.svm(Veracity ~. ,data = train_raw.df,
#                           kernel = "sigmoid",
#                           gamma = c(0.1,0.5,1,2,3,4),
#                           coef0 = c(0.1,0.5,1,2,3,4))
# summary(sigmoid.tune)
# best.sigmoid <- sigmoid.tune$best.model
# sigmoid.test <- predict(best.sigmoid, test_raw.df)
# table(sigmoid.test,test_raw.df$Veracity)
# confusionMatrix(sigmoid.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference")) # poor 64%, k



# tuning best svm model for polynomial kernel
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 3,coef0 = 3,  misclassification rate no larger than 17%( better than
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       3     2
##
## - best performance: 0.1801538
##
## - Detailed performance results:
##    degree coef0     error dispersion
## 1       2   0.1 0.1995385 0.09488995
## 2       3   0.1 0.2347692 0.06855012
## 3       4   0.1 0.2540000 0.08653594
## 4       5   0.1 0.2616923 0.07336184
## 5       6   0.1 0.3007692 0.11554348
## 6       2   0.5 0.1881538 0.06767307
## 7       3   0.5 0.1838462 0.06148823
## 8       4   0.5 0.2112308 0.05083420
## 9       5   0.5 0.2496923 0.07873774
## 10      6   0.5 0.2378462 0.06759978
## 11      2   1.0 0.1883077 0.06603727
## 12      3   1.0 0.1918462 0.07361449
## 13      4   1.0 0.2384615 0.05648480
## 14      5   1.0 0.2343077 0.09723610
## 15      6   1.0 0.2575385 0.09320702
## 16      2   2.0 0.1961538 0.07161241
## 17      3   2.0 0.1801538 0.07115262
```

```
## 18      4   2.0 0.2426154 0.10952542
## 19      5   2.0 0.2615385 0.11456920
## 20      6   2.0 0.2536923 0.11897977
## 21      2   3.0 0.1921538 0.06740206
## 22      3   3.0 0.1916923 0.06860381
## 23      4   3.0 0.2232308 0.10888340
## 24      5   3.0 0.2420000 0.12314752
## 25      6   3.0 0.2420000 0.12314752
## 26      2   4.0 0.1921538 0.06740206
## 27      3   4.0 0.1956923 0.07314050
## 28      4   4.0 0.2033846 0.08356866
## 29      5   4.0 0.2307692 0.10987052
## 30      6   4.0 0.2420000 0.12314752
```

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

```
##
## poly.test  0  1
##         0 29 11
##         1  3 21
```

```
confusionMatrix(poly.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference")) # 78.12% accuracy,
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29 11
##          1  3 21
##
##                Accuracy : 0.7812
##                  95% CI : (0.6603, 0.8749)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 3.535e-06
##
##                   Kappa : 0.5625
##
##  Mcnemar's Test P-Value : 0.06137
##
##             Sensitivity : 0.9062
##             Specificity : 0.6562
##          Pos Pred Value : 0.7250
##          Neg Pred Value : 0.8750
##              Prevalence : 0.5000
##          Detection Rate : 0.4531
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.7812
##
##        'Positive' Class : 0
##
```

```
# tuning best svm model for radial kernel
rad.tune <- tune.svm(Veracity ~ ., data = train_raw.df,
                      kernel = "radial",
```

```r
                      gamma = c(0.1, 0.5, 1, 2, 3, 4))
summary(rad.tune) #best gamma = 0.1,  misclassification rate no larger than 19%( better than linear ker
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma
##    0.1
##
## - best performance: 0.2104615
##
## - Detailed performance results:
##   gamma     error dispersion
## 1   0.1 0.2104615 0.07648353
## 2   0.5 0.2730769 0.05569830
## 3   1.0 0.3964615 0.15243119
## 4   2.0 0.5301538 0.12686106
## 5   3.0 0.5463077 0.11871867
## 6   4.0 0.5580000 0.10216602
```

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

```
##
## rad.test  0  1
##        0 26 15
##        1  6 17
```

```r
confusionMatrix(rad.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference")) # 79% accuracy, kap
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26 15
##          1  6 17
##
##                Accuracy : 0.6719
##                  95% CI : (0.5431, 0.7841)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.004073
##
##                   Kappa : 0.3438
##
##  Mcnemar's Test P-Value : 0.080856
##
##             Sensitivity : 0.8125
##             Specificity : 0.5312
##          Pos Pred Value : 0.6341
##          Neg Pred Value : 0.7391
##              Prevalence : 0.5000
```

```
##          Detection Rate : 0.4062
##    Detection Prevalence : 0.6406
##       Balanced Accuracy : 0.6719
##
##        'Positive' Class : 0
##
```

```r
# feature extraction
set.seed(3117)
rfeCNTL <- rfeControl(functions = lrFuncs, method = "cv", number = 11)
svm.features <- rfe(train_raw.df[,1:11], train_raw.df[,12],
                    sizes = c(11, 10, 9, 8, 7, 6, 5),
                    rfeControl = rfeCNTL,
                    method = "svmLinear")
svm.features
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (11 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          5   0.7267 0.4527    0.11186  0.2226
##          6   0.7346 0.4685    0.10007  0.1989        *
##          7   0.7227 0.4449    0.09957  0.1980
##          8   0.7226 0.4448    0.10679  0.2125
##          9   0.7269 0.4526    0.10696  0.2139
##         10   0.7265 0.4521    0.11128  0.2215
##         11   0.7226 0.4444    0.11156  0.2222
##
## The top 5 variables (out of 6):
##    Accuracy, TruthProp, Valence, Race, WordCount
```

```r
svm.features$fit$coefficients # Accuracy, TruthProp, VidLength_ms, VidLength_sec, Valence
```

```
##  (Intercept)     Accuracy    TruthProp      Valence         Race    WordCount
##  0.002390685  1.202324466  0.535337640 -0.193218718  0.151619361 -0.149551557
##      Anxious
##  0.154003445
```

```r
# use above 8 features to train polynomial svm

#SVM split train and test 80/20

train_raw.df <- as.data.frame(MU3D_Video_Level_Data.scaled[train_ind_raw, c(1,4,5,7,8,12)])
test_raw.df <- as.data.frame(MU3D_Video_Level_Data.scaled[-train_ind_raw, c(1,4,5,7,8,12)])
levels <- unique(c(train_raw.df$Veracity, test_raw.df$Veracity))
test_raw.df$Veracity  <- factor(test_raw.df$Veracity, levels=levels)
train_raw.df$Veracity <- factor(train_raw.df$Veracity, levels=levels)
#write.csv(test_raw.df, "test_raw.df.csv")
#write.csv(train_raw.df,"train_raw.df.csv")

# tuning best svm model for polynomial kernel
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 = 4,  misclassification rate no larger than 19%( better than
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       5     4
##
## - best performance: 0.074
##
## - Detailed performance results:
##    degree coef0      error dispersion
## 1       2   0.1 0.15984615 0.06866224
## 2       3   0.1 0.17169231 0.04461951
## 3       4   0.1 0.17953846 0.07277012
## 4       5   0.1 0.18630769 0.10956000
## 5       6   0.1 0.20584615 0.10446646
## 6       2   0.5 0.14846154 0.06256936
## 7       3   0.5 0.17184615 0.05561231
## 8       4   0.5 0.15184615 0.06311256
## 9       5   0.5 0.16769231 0.06501707
## 10      6   0.5 0.17169231 0.06551350
## 11      2   1.0 0.14861538 0.05141112
## 12      3   1.0 0.14815385 0.04645514
## 13      4   1.0 0.15200000 0.06679199
## 14      5   1.0 0.15584615 0.06980785
## 15      6   1.0 0.15969231 0.07073875
## 16      2   2.0 0.14461538 0.04843915
## 17      3   2.0 0.13615385 0.06799950
## 18      4   2.0 0.15984615 0.04857712
## 19      5   2.0 0.13676923 0.06656185
## 20      6   2.0 0.11353846 0.03923529
## 21      2   3.0 0.14461538 0.04843915
## 22      3   3.0 0.14015385 0.06545828
## 23      4   3.0 0.16015385 0.05018545
## 24      5   3.0 0.10553846 0.04133982
## 25      6   3.0 0.07815385 0.05208500
## 26      2   4.0 0.14461538 0.04843915
## 27      3   4.0 0.14061538 0.05526197
## 28      4   4.0 0.13676923 0.06736303
## 29      5   4.0 0.07400000 0.05254571
## 30      6   4.0 0.09000000 0.05237929
```

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

```
##
```

```
## poly.test  0  1
##         0 29  1
##         1  3 31
```

```
confusionMatrix(poly.test, test_raw.df$Veracity, dnn = c("Prediction", "Reference")) # 95.31% accuracy,
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  1
##          1  3 31
##
##                Accuracy : 0.9375
##                  95% CI : (0.8476, 0.9827)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 3.682e-14
##
##                   Kappa : 0.875
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9062
##             Specificity : 0.9688
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.9118
##              Prevalence : 0.5000
##          Detection Rate : 0.4531
##    Detection Prevalence : 0.4688
##       Balanced Accuracy : 0.9375
##
##        'Positive' Class : 0
##
```