Audio analysis: 8 credits

Introduction of the assignment: - Purpose: to distinguish emergency vehicles from general traffic. - Data: 600 sound samples from Bose, including200 of ambulances (1-200), 200 of fire trucks (201-400) and 200 of general traffic (401-600). However, I will use 300.

```
# Install necessary libraries
library(soundgen)
library(seewave)
library(tuneR)
library(caret)
library(e1071)
library(MLmetrics)
library(randomForest)
```

Question 12: data loading and feature extraction

Now, we use readWave() to read in the audio files into R.

```
# Produce character vectors of the names of files in the named directory
ambulanceFiles <- list.files(path = "ambulance", pattern = ".wav", full.names = T)
fireTruckFiles <- list.files(path = "firetruck", pattern = ".wav", full.names = T)
trafficFiles <- list.files(path = "traffic", pattern = ".wav", full.names = T)

# Choose a random subset of 75 files (for each of ambulance and fire truck) and 150 files (for general
set.seed(1010)
selectedAmbulance <- sample(ambulanceFiles, 75)
selectedFireTruck <- sample(fireTruckFiles, 75)
selectedTraffic <- sample(trafficFiles, 150)

# Apply readWave for all the files to create 2 datasets: Emergency and NotEmergency
df_Emergency <- lapply(c(selectedAmbulance, selectedFireTruck), FUN = function (f) readWave(f))
df_NotEmergency <- lapply(selectedTraffic, FUN = function (f) readWave(f))

# See the dataset length
length(df_Emergency)
```

```
## [1] 150
```

```
length(df_NotEmergency)
```

```
## [1] 150
```

Here, I extract the fundamental frequency of each sound file using the fund(), including two features based on the fundamental frequency: The average frequency across the clip, and the standard deviation. Omit NA values from your calculations.

```
## For Emergency dataset

fundaFreqEmergency <- lapply(df_Emergency, FUN = function(f) fund(f, plot = FALSE))

# Omit NA values
```

```r
fundaFreqEmergency2 <- lapply(fundaFreqEmergency, FUN = function(f) na.omit(f))
rm(fundaFreqEmergency)

# Omit the sounds without any non-NA fundamental frequencies in the fundamental frequency list
fundaFreqEmergency3 <- fundaFreqEmergency2[sapply(fundaFreqEmergency2, function(f) (nrow(f) > 0))]

# Now look at the number of sounds left
length(fundaFreqEmergency3)
```

```
## [1] 145
```

```r
# Create a matrix with the the first column of 1 (1 = emergency vehicle)
df_Emergency2 <- matrix(1, nrow = length(fundaFreqEmergency3), ncol = 1)

# Calculate average frequency and standard deviation and merge it to newly created data set
df_Emergency2 <- data.frame(cbind(df_Emergency2, t(vapply(fundaFreqEmergency3, FUN = function(f) c(meanF
head(df_Emergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq
## 1  1      9.386249    7.246295
## 2  1      3.022435    4.801456
## 3  1      8.104066    6.985160
## 4  1      8.726577    7.786374
## 5  1      7.096138    5.732495
## 6  1      7.586933    6.599582
```

```r
dim(df_Emergency2)
```

```
## [1] 145   3
```

```r
## Do similarly for NotEmergency dataset

fundaFreqNotEmergency <- lapply(df_NotEmergency, FUN = function(f) fund(f, plot = FALSE))

# Omit NA values
fundaFreqNotEmergency2 <- lapply(fundaFreqNotEmergency, FUN = function(f) na.omit(f))
rm(fundaFreqNotEmergency)

# Omit the sounds without any non-NA fundamental frequencies in the fundamental frequency list
fundaFreqNotEmergency3 <- fundaFreqNotEmergency2[sapply(fundaFreqNotEmergency2, function(f) (nrow(f) > 0

# Now look at the number of sounds left
length(fundaFreqNotEmergency3)
```

```
## [1] 150
```

```r
# Create a matrix with the the first column of 0 (0 = emergency vehicle)
df_NotEmergency2 <- matrix(0, nrow = length(fundaFreqNotEmergency3), ncol = 1)

# Calculate average frequency and standard deviation and merge it to newly created data set
df_NotEmergency2 <- data.frame(cbind(df_NotEmergency2, t(vapply(fundaFreqNotEmergency3, FUN = function(
head(df_NotEmergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq
## 1  0      6.458885    5.142417
## 2  0      7.031167    6.099147
## 3  0      6.252602    5.553349
## 4  0      6.834510    5.678362
## 5  0      6.695009    5.068726
## 6  0      6.357217    5.381736
```

```
dim(df_NotEmergency2)
```

```
## [1] 150   3
```

The second feature I will extract is jitter, the average absolute difference in fundamental frequency across the clip. I will compute this using the fund() function.

```
## For Emergency dataset

# Create a function to calculate average absolute difference, given y as the index of column
CalculateAveAbsDiff <- function(x, y) {
  sum <- 0
  if (nrow(x) == 1) {
    return(x[1, y])
  }
  else {
  for (i in 1:(nrow(x)-1)) {
    sum <- sum + abs((x[i, y] - x[i + 1, y]))
  }
  return(sum/(nrow(x)-1))
  }
}

# Calculate jitter and merge it
df_Emergency2$jitter <- sapply(fundaFreqEmergency3, CalculateAveAbsDiff, y=2)
head(df_Emergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq   jitter
## 1  1      9.386249    7.246295 0.000000
## 2  1      3.022435    4.801456 4.811446
## 3  1      8.104066    6.985160 0.000000
## 4  1      8.726577    7.786374 0.000000
## 5  1      7.096138    5.732495 1.225000
## 6  1      7.586933    6.599582 1.837500
```

```
## Do similarly for NotEmergency dataset

# Calculate jitter and merge it
df_NotEmergency2$jitter <- sapply(fundaFreqNotEmergency3, CalculateAveAbsDiff, y=2)
head(df_NotEmergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq    jitter
## 1  0      6.458885    5.142417 0.8647059
## 2  0      7.031167    6.099147 2.5442308
## 3  0      6.252602    5.553349 0.3195652
## 4  0      6.834510    5.678362 1.6704545
## 5  0      6.695009    5.068726 1.0500000
## 6  0      6.357217    5.381736 1.0500000
```

Next, I will extract shimmer. Shimmer is defined as the average absolute difference in amplitude across the clip. I will use the env() function with envt = "abs" to extract the amplitude across the clip, and compute the shimmer.

```
## For Emergency dataset

amplitudeEmergency <- lapply(df_Emergency, FUN = function(f)
  env(f, envt = "abs", plot = FALSE))

# Calculate shimmer and merge it
df_Emergency2$shimmer <- sapply(amplitudeEmergency, CalculateAveAbsDiff, y = 1)[sapply(fundaFreqEmergen

head(df_Emergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq   jitter   shimmer
## 1  1      9.386249    7.246295 0.000000 2157.7320
## 2  1      3.022435    4.801456 4.811446  697.1377
## 3  1      8.104066    6.985160 0.000000 1149.2370
## 4  1      8.726577    7.786374 0.000000  877.5498
## 5  1      7.096138    5.732495 1.225000  638.4037
## 6  1      7.586933    6.599582 1.837500  561.0140
```

```
## Do similarly for NotEmergency dataset

amplitudeNotEmergency <- lapply(df_NotEmergency, FUN = function(f)
  env(f, envt = "abs", plot = FALSE))

# Calculate shimmer and merge it
df_NotEmergency2$shimmer <- sapply(amplitudeNotEmergency, CalculateAveAbsDiff, y = 1)[sapply(fundaFreqN

head(df_NotEmergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq    jitter  shimmer
## 1  0      6.458885    5.142417 0.8647059 168.5867
## 2  0      7.031167    6.099147 2.5442308 170.6853
## 3  0      6.252602    5.553349 0.3195652 326.6149
## 4  0      6.834510    5.678362 1.6704545 196.9244
## 5  0      6.695009    5.068726 1.0500000 160.5970
## 6  0      6.357217    5.381736 1.0500000 164.3510
```

Also, I will extract harmonic-to-noise ratio, pitch, loudness, and timbre. Using soundgen library, I will extract harmonic-to-noise ratio (mean and standard deviation), pitch (mean and standard deviation), loudness (mean and standard deviation), and timbre (as specCentroid, mean and standard deviation).

```
## For Emergency dataset

CalculateFeatures <- function (x) {
  return(analyze(x, pitchMethods = "autocor")$summary[, c("HNR_mean", "HNR_sd", "pitch_mean", "pitch_sd
}

df_Emergency2 <- data.frame(cbind(df_Emergency2,t(sapply(df_Emergency[sapply(fundaFreqEmergency2, functi

head(df_Emergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq   jitter   shimmer HNR_mean   HNR_sd pitch_mean
## 1  1      9.386249    7.246295 0.000000 2157.7320 19.21383 3.151797    1411.76
## 2  1      3.022435    4.801456 4.811446  697.1377 21.29872 6.123902   1027.039
## 3  1      8.104066    6.985160 0.000000 1149.2370 15.37705  2.56805   1128.986
## 4  1      8.726577    7.786374 0.000000  877.5498 9.731571 4.947918   1425.495
## 5  1      7.096138    5.732495 1.225000  638.4037 11.62778 3.079564   867.6503
## 6  1      7.586933    6.599582 1.837500  561.0140 7.331616 2.637203   1250.451
##   pitch_sd loudness_mean loudness_sd specCentroidVoiced_mean
## 1 259.4477      11.24685    1.301929                1992.002
## 2 173.3206      6.951284    0.929473                1946.655
## 3 197.9829      9.447234   0.9739675                1515.666
## 4 154.7107      11.13007    2.612156                1598.022
## 5 151.0153      12.41051    1.974857                1525.893
## 6 140.4648       7.94056   0.8060171                1984.971
##   specCentroidVoiced_sd
## 1              313.6087
## 2               162.314
## 3              153.7959
## 4              198.1509
## 5              274.7944
## 6              105.1301
```

```
## Do similarly for NotEmergency dataset

df_NotEmergency2 <- data.frame(cbind(df_NotEmergency2,t(sapply(df_NotEmergency[sapply(fundaFreqNotEmerge

head(df_NotEmergency2)
```

```
##   V1 meanFundaFreq sdFundaFreq    jitter  shimmer HNR_mean   HNR_sd pitch_mean
## 1  0      6.458885    5.142417 0.8647059 168.5867 5.782686 2.054135         NA
## 2  0      7.031167    6.099147 2.5442308 170.6853 6.732252 2.026629    299.331
## 3  0      6.252602    5.553349 0.3195652 326.6149 1.152516 1.976493   510.4502
## 4  0      6.834510    5.678362 1.6704545 196.9244 3.260842 2.009322   1998.402
## 5  0      6.695009    5.068726 1.0500000 160.5970 5.163598 2.244271   1075.186
## 6  0      6.357217    5.381736 1.0500000 164.3510 6.016551 2.109381         NA
##   pitch_sd loudness_mean loudness_sd specCentroidVoiced_mean
## 1       NA      7.418725   0.4386759                      NA
## 2 569.1203       8.37587   0.9869734                918.7526
## 3 569.1871      10.03458    2.139419                1553.823
## 4 502.1023      7.552399   0.3876096                1360.474
## 5       NA      7.041417   0.4534585                1058.797
## 6       NA      7.605968   0.4685794                      NA
##   specCentroidVoiced_sd
## 1                    NA
## 2              128.9416
## 3              236.2393
## 4              138.8458
## 5                    NA
## 6                    NA
```

I have already created a binary variable (1 = emergency vehicle, 0 = general traffic) that can serve as my prediction target and add it to 2 datasets. Now, I will combine 2 datasets before splitting them to train-test set.

```r
df_Sound <- rbind(df_Emergency2, df_NotEmergency2)
colnames(df_Sound)[1] <- "source"
```

Then I will split it to 80% for training, 20% for testing, with balanced classes in both dataset.

```r
# convert source variable into factor variable
df_Sound$source <- as.factor(df_Sound$source)

# convert all independent variables into numeric
df_Sound[, c(2:13)] <- sapply(df_Sound[, c(2:13)], as.numeric)

# Omit rows with NA values
df_Sound <- na.omit(df_Sound)

# train-test data split using caret library
set.seed(1015)
trainIndex <- createDataPartition(df_Sound$source, p = .8, list = FALSE, times = 1)
train <- df_Sound[trainIndex,]
test <- df_Sound[-trainIndex,]
xTrain <- subset(train, select = -source)
yTrain <- subset(train, select = source)
xTest <- subset(test, select = -source)
yTest <- subset(test, select = source)
```

Question 13: Prediction model

Here 3 different models will be used to predict the source of sound based on extracted acoustic features: logistic regression model, support vector machine (SVM), and random forest (RF).

```r
# Logistic classification model for source using glm()
modelLogit <- glm(formula = source ~ .,
                  family = binomial(link = "logit"),
                  data = train)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Make prediction (test)
yPredLogitTest <- predict(modelLogit, xTest, type="response")
yPredLogitTest <- ifelse(yPredLogitTest>0.5, 1, 0)

# Precision, Recall, Accuracy, F1-Score for source classification (test)
```

```
df_PerformanceLogitTest <- list(data.frame(
  precision = Precision(yTest$source,
                         yPredLogitTest,
                         positive = 1),
  recall = Recall(yTest$source,
                   yPredLogitTest,
                   positive = 1),
  accuracy = Accuracy(yTest$source,
                       yPredLogitTest),
  f1Score = F1_Score(yTest$source,
                       yPredLogitTest)),
  confusionMatrix(table(yPredLogitTest, yTest$source))
)

df_PerformanceLogitTest
```

```
## [[1]]
##   precision recall accuracy f1Score
## 1         1      1        1       1
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredLogitTest  0  1
##              0 19  0
##              1  0 27
##
##                Accuracy : 1
##                  95% CI : (0.9229, 1)
##     No Information Rate : 0.587
##     P-Value [Acc > NIR] : 2.269e-11
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.413
##          Detection Rate : 0.413
##    Detection Prevalence : 0.413
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
# Make prediction (train)
yPredLogitTrain <- predict(modelLogit, xTrain, type="response")
yPredLogitTrain <- ifelse(yPredLogitTrain>0.5, 1, 0)
```

```
# Precision, Recall, Accuracy, F1-Score for source classification (test)
df_PerformanceLogitTrain <- list(data.frame(
  precision = Precision(yTrain$source,
                        yPredLogitTrain,
                        positive = 1),
  recall = Recall(yTrain$source,
                  yPredLogitTrain,
                  positive = 1),
  accuracy = Accuracy(yTrain$source,
                      yPredLogitTrain),
  f1Score = F1_Score(yTrain$source,
                      yPredLogitTrain)),
  confusionMatrix(table(yPredLogitTrain, yTrain$source))
)

df_PerformanceLogitTrain
```

```
## [[1]]
##   precision recall accuracy f1Score
## 1         1      1        1       1
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredLogitTrain   0   1
##               0  77   0
##               1   0 108
##
##                Accuracy : 1
##                  95% CI : (0.9803, 1)
##     No Information Rate : 0.5838
##     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 : 0.4162
##          Detection Rate : 0.4162
##    Detection Prevalence : 0.4162
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
# Linear SVM classification model for source using e1071 library
modelSVMLinear <- svm(source ~ .,
                  data = train,
                  type = 'C-classification',
                  kernel = 'linear')


# Make prediction (test)
yPredSVMLinearTest <- predict(modelSVMLinear, xTest)

# Precision, Recall, Accuracy, F1-Score for source classification (test)
df_PerformanceSVMLinearTest <- list(data.frame(
  precision = Precision(yTest$source,
                        yPredSVMLinearTest,
                        positive = 1),
  recall = Recall(yTest$source,
                  yPredSVMLinearTest,
                  positive = 1),
  accuracy = Accuracy(yTest$source,
                      yPredSVMLinearTest),
  f1Score = F1_Score(yTest$source,
                      yPredSVMLinearTest)),
  confusionMatrix(table(yPredSVMLinearTest, yTest$source))
)

df_PerformanceSVMLinearTest
```

```
## [[1]]
##   precision   recall  accuracy  f1Score
## 1         1 0.962963 0.9782609 0.974359
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredSVMLinearTest  0  1
##                  0 19  1
##                  1  0 26
##
##                Accuracy : 0.9783
##                  95% CI : (0.8847, 0.9994)
##     No Information Rate : 0.587
##     P-Value [Acc > NIR] : 7.572e-10
##
##                   Kappa : 0.9555
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9630
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4130
##          Detection Rate : 0.4130
##    Detection Prevalence : 0.4348
##       Balanced Accuracy : 0.9815
```

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

```
# Make prediction (train)
yPredSVMLinearTrain <- predict(modelSVMLinear, xTrain)

# Precision, Recall, Accuracy, F1-Score for source classification (test)
df_PerformanceSVMLinearTrain <- list(data.frame(
  precision = Precision(yTrain$source,
                         yPredSVMLinearTrain,
                         positive = 1),
  recall = Recall(yTrain$source,
                   yPredSVMLinearTrain,
                   positive = 1),
  accuracy = Accuracy(yTrain$source,
                       yPredSVMLinearTrain),
  f1Score = F1_Score(yTrain$source,
                       yPredSVMLinearTrain)),
  confusionMatrix(table(yPredSVMLinearTrain, yTrain$source))
)

df_PerformanceSVMLinearTrain
```

```
## [[1]]
##   precision    recall  accuracy   f1Score
## 1         1 0.9907407 0.9945946 0.9935484
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredSVMLinearTrain   0   1
##                   0  77   1
##                   1   0 107
##
##                Accuracy : 0.9946
##                  95% CI : (0.9703, 0.9999)
##     No Information Rate : 0.5838
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9889
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9907
##          Pos Pred Value : 0.9872
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4162
##          Detection Rate : 0.4162
##    Detection Prevalence : 0.4216
##       Balanced Accuracy : 0.9954
##
```

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

```
# Random Forest classification model for source of sound

modelRF <- randomForest(source ~ .,
                        data = train,
                        importance = TRUE,
                        proximity = TRUE)



# Make prediction (Test)
yPredRFTest <- predict(modelRF, xTest)

# Precision, Recall, Accuracy, F1-Score for source classification (Test)
df_PerformanceRFTest <- list(data.frame(
  precision = Precision(yTest$source,
                         yPredRFTest,
                        positive = 1),
  recall = Recall(yTest$source,
                   yPredRFTest,
                  positive = 1),
  accuracy = Accuracy(yTest$source,
                       yPredRFTest),
  f1Score = F1_Score(yTest$source,
                      yPredRFTest)
), confusionMatrix(table(yPredRFTest, yTest$source)))

df_PerformanceRFTest
```

```
## [[1]]
##   precision   recall  accuracy  f1Score
## 1         1 0.962963 0.9782609 0.974359
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredRFTest  0  1
##           0 19  1
##           1  0 26
##
##                Accuracy : 0.9783
##                  95% CI : (0.8847, 0.9994)
##     No Information Rate : 0.587
##     P-Value [Acc > NIR] : 7.572e-10
##
##                   Kappa : 0.9555
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9630
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4130
##          Detection Rate : 0.4130
##    Detection Prevalence : 0.4348
##       Balanced Accuracy : 0.9815
##
##        'Positive' Class : 0
##
```

```
# Make prediction (Train)
yPredRFTrain <- predict(modelRF, xTrain)

# Precision, Recall, Accuracy, F1-Score for source classification (Train)
df_PerformanceRFTrain <- list(data.frame(
  precision = Precision(yTrain$source,
                         yPredRFTrain,
                        positive = 1),
  recall = Recall(yTrain$source,
                   yPredRFTrain,
                  positive = 1),
  accuracy = Accuracy(yTrain$source,
                       yPredRFTrain),
  f1Score = F1_Score(yTrain$source,
                      yPredRFTrain)
), confusionMatrix(table(yPredRFTrain, yTrain$source)))

df_PerformanceRFTrain
```

```
## [[1]]
##   precision recall accuracy f1Score
## 1         1      1        1       1
##
## [[2]]
## Confusion Matrix and Statistics
##
##
## yPredRFTrain   0   1
##            0  77   0
##            1   0 108
##
##                Accuracy : 1
##                  95% CI : (0.9803, 1)
##     No Information Rate : 0.5838
##     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 : 0.4162
##          Detection Rate : 0.4162
##    Detection Prevalence : 0.4162
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

Question 14: Result interpretation for logit model

```
summary(modelLogit)
```

```
##
## Call:
## glm(formula = source ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.198e-05  -2.100e-08   2.100e-08   2.100e-08   7.049e-05
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.781e+02  2.244e+05  -0.001    0.999
## meanFundaFreq            7.156e+00  4.765e+04   0.000    1.000
## sdFundaFreq              7.657e+00  1.023e+05   0.000    1.000
## jitter                   7.690e+00  3.165e+04   0.000    1.000
## shimmer                  2.690e-02  2.142e+02   0.000    1.000
## HNR_mean                 5.130e+00  6.027e+03   0.001    0.999
## HNR_sd                  -1.848e+00  6.156e+04   0.000    1.000
## pitch_mean               5.683e-03  2.944e+01   0.000    1.000
## pitch_sd                -3.735e-02  1.137e+02   0.000    1.000
## loudness_mean           -7.131e+00  1.643e+04   0.000    1.000
## loudness_sd              2.413e+01  6.558e+04   0.000    1.000
## specCentroidVoiced_mean  1.543e-01  1.083e+02   0.001    0.999
## specCentroidVoiced_sd   -1.883e-01  6.455e+02   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.5125e+02  on 184  degrees of freedom
## Residual deviance: 1.5358e-08  on 172  degrees of freedom
## AIC: 26
##
## Number of Fisher Scoring iterations: 25
```

Interestingly, although the logit model is perfect for prediction (see question 15 later), when we look into which variable affects the prediction ability through this table, we do not see any significant p-value (below 0.05). However, this is not due to the lack of variable that affects the prediction ability. Because the sample is not large (around 185 for train data), including 12 predictors is too much and they can be correlated with each other, which results in no significant coefficient.

To see which variables significantly affect prediction ability and in which direction, I will run logit model 12 times. In each time, only one predictor is included. This will shed light on the signficance of each variable.

```r
for (x in c("meanFundaFreq", "sdFundaFreq", "jitter", "shimmer", "HNR_mean", "HNR_sd",
            "pitch_mean", "pitch_sd", "loudness_mean", "loudness_sd",
            "specCentroidVoiced_mean", "specCentroidVoiced_sd")) {
  print(summary(glm(formula = paste("source ~ ", x ), family = binomial(link = 'logit'), data = train))$
}
```

```
##                  Estimate Std. Error    z value  Pr(>|z|)
## (Intercept)   -0.24263519  0.9399840 -0.2581269 0.7963090
## meanFundaFreq  0.08557546  0.1368441  0.6253499 0.5317415
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -3.5952305  1.0818786 -3.323137 0.0008901123
## sdFundaFreq  0.6790949  0.1877128  3.617733 0.0002971947
##               Estimate Std. Error    z value    Pr(>|z|)
## (Intercept) -0.2347541  0.2377555 -0.9873762 0.323458235
## jitter       0.3360036  0.1238722  2.7125018 0.006677742
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -7.44066734 1.45746324 -5.105218 3.304136e-07
## shimmer      0.02634514 0.00636469  4.139265 3.484207e-05
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -1.171507 0.30722207 -3.813226 1.371648e-04
## HNR_mean     0.253546 0.05016191  5.054553 4.313994e-07
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -7.561615  1.2235143 -6.180242 6.400322e-10
## HNR_sd       2.848172  0.4777885  5.961156 2.504596e-09
##                 Estimate   Std. Error   z value    Pr(>|z|)
## (Intercept)  1.362802457 0.4369012299  3.119246 0.001813145
## pitch_mean  -0.000996227 0.0003943763 -2.526082 0.011534254
##                  Estimate   Std. Error    z value   Pr(>|z|)
## (Intercept)  0.4887383351 0.2239566886  2.1822895 0.02908817
## pitch_sd    -0.0006792519 0.0007499751 -0.9056994 0.36509500
##                 Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)   -5.1447279 0.84104932 -6.117035 9.533258e-10
## loudness_mean  0.6133611 0.09843994  6.230816 4.640129e-10
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -2.319965  0.4175692 -5.555882 2.762130e-08
## loudness_sd  2.757740  0.4609766  5.982386 2.198926e-09
##                             Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)             -18.97076623 3.496356974 -5.425867 5.767405e-08
## specCentroidVoiced_mean   0.01307859 0.002427477  5.387730 7.135319e-08
##                          Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)           -3.35138965 0.532386644 -6.295030 3.073413e-10
## specCentroidVoiced_sd  0.02238612 0.003277158  6.830955 8.435122e-12
```

It is easy to see that sdFundaFreq, jitter, shimmer, HNR_mean, HNR_sd, pitch_mean, loudness_mean, loudness_sd, specCentroidVoiced_mean, specCentroidVoiced_sd are variables that have statistically significant impact on dependent variable, as their P-values are below 0.05. Meanwhile, mean of fundamental frequency and mean of pitch have no statistically significant impact on the prediction.

Among them, 2 variables with the most considerable coefficients are HNR_sd (2.85) and loudness_sd (2.76), which are standard deviation of harmonic to noise ratio and standard deviation of loudness. This is understandable because it is very rare for general traffic to resemble the high standard deviation of loudness in ambulance and fire truck siren.

Question 15: Model comparison

It is surprising that all the 3 models perform perfectly or almost perfectly in predicting which sound is from general traffic and which is from emergency vehicles for the test set.

Among them, logit model is the best one, with accuracy, recall, precision, F1-score as 1. Following that are Linear SVM and Random Forest, with accuracy of 0.98 and recall of 0.96. This is corresponding to 1 case of emergency vehicles but the models fail to predict. Nevertheless, the precision for both models is still perfect at 1, which means all the emergency prediction is accurate.

This result is also due to the fact that relatively sufficient acoustic features have been extracted from the sound files, and there are many distinct differences between sounds from general traffic and sounds from emergency vehicles. To test which one will perform really consistently in out-of-sample data, it is necessary to include different sound files with more diverse acoustic features. For example, ambulance or fire truck siren sounds can be recorded in different distances from the sources, which reflects real-life scenario in which the noise-cancellation function in a headset should always perform properly. This will result in higher variety of acoustic features, which in turn makes it challenging for machine learning models to linearly predict the sources.

Temporarily, for the purpose of embedding noise cancellation in Bose headset, any of the 3 models proves to be good enough to be selected. The best model is the logit model, as it has perfect recall compared to the other two models. It means headset users will not miss any single emergency sound signals from their surrounding environment, ensuring their safety. Besides, all the 3 models will guarantee comfort for users, as every time the headsets mute themselves, the users will realize it is a correct decision (precision = 1). The users will feel annoyed if the their headsets mute themselves when there is no emergency sounds outside, which is not observed in the models for test set here.