Tree based methods

Pia Glimmerfors

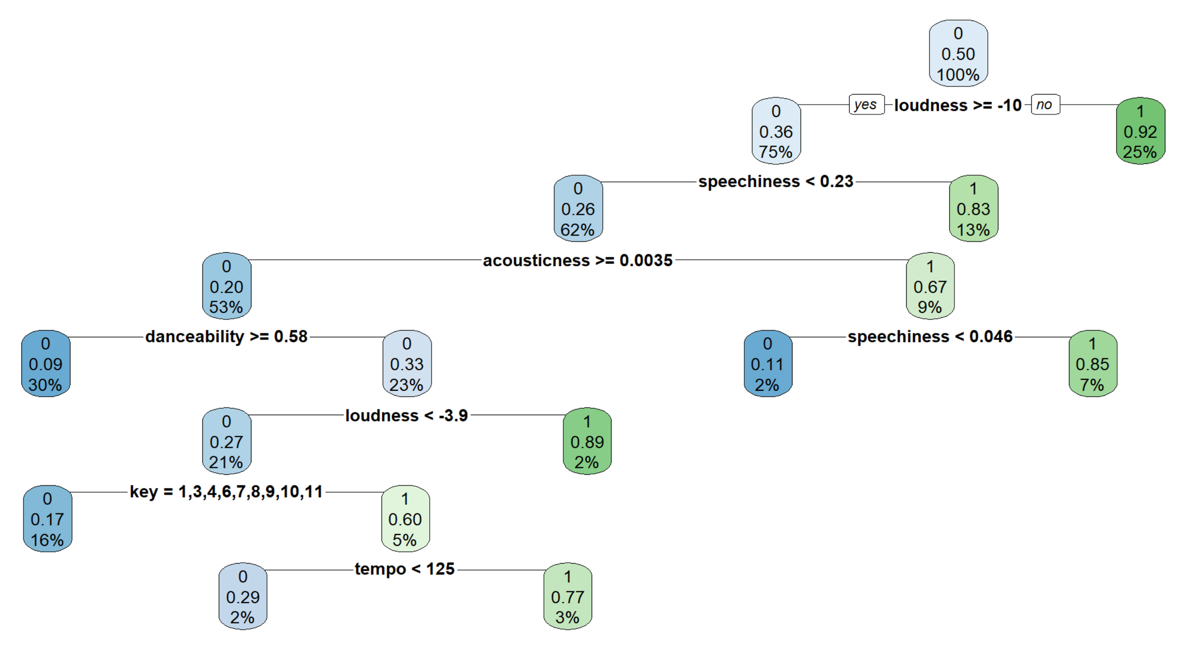
2024-10-16

Loading data, fixing misslabled values, splitting into test and training sets

```r
test_df <- read.csv("project_test.csv")
train_df <- read.csv("project_train.csv")

# Change values in misslabled rows
train_df$energy[85] <- 7.34e-02
train_df$loudness[95] <- -6.542

# Change from int to factor
test_df$key <- factor(test_df$key)
test_df$mode <- factor(test_df$mode)
train_df$key <- factor(train_df$key)
train_df$mode <- factor(train_df$mode)
train_df$Label <- factor(train_df$Label)

# Split into train and test data
set.seed(32)
data_split <- initial_split(train_df, prop = 0.8)
train_data <- training(data_split)
test_data <- testing(data_split)
```

Decision tree

```r
set.seed(32)

model_tree <- rpart(Label ~., data = train_data)  # Constructing decision tree
pred_tree <- predict(model_tree, test_data, type = "class") # Making predictions on test data

rpart.plot(model_tree, fallen.leaves = FALSE)  # Plot decision tree
```




```r
confusionMatrix(pred_tree, test_data$Label) # Plot confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 37 16
##          1 13 35
##
##                Accuracy : 0.7129
##                  95% CI : (0.6143, 0.7985)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : 1.724e-05
##
##                   Kappa : 0.426
##
##  Mcnemar's Test P-Value : 0.7103
##
##             Sensitivity : 0.7400
##             Specificity : 0.6863
##          Pos Pred Value : 0.6981
##          Neg Pred Value : 0.7292
##              Prevalence : 0.4950
##          Detection Rate : 0.3663
##    Detection Prevalence : 0.5248
##       Balanced Accuracy : 0.7131
##
##        'Positive' Class : 0
##
```

Random forest

```r
set.seed(32)

# Constructing random forest
model_rf <- randomForest(x = train_data[,-12], y = train_data[,12], xtest = test_data[,-12], ytest = test_data[,12],
                         type = "classification")

confusionMatrix(model_rf$test$predicted, test_data$Label)  # Print confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41 14
##          1  9 37
##
##                Accuracy : 0.7723
##                  95% CI : (0.6782, 0.8498)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : 3.07e-08
##
##                   Kappa : 0.545
##
##  Mcnemar's Test P-Value : 0.4042
##
##             Sensitivity : 0.8200
##             Specificity : 0.7255
##          Pos Pred Value : 0.7455
##          Neg Pred Value : 0.8043
##              Prevalence : 0.4950
##          Detection Rate : 0.4059
##    Detection Prevalence : 0.5446
##       Balanced Accuracy : 0.7727
##
##        'Positive' Class : 0
##
```

Gradient Boosting

```r
set.seed(32)

train_data_gbm <- train_data %>%
  mutate(Label = as.numeric(Label)) %>%
  mutate(Label = if_else(Label == 1, 0, 1))

# Constructing gradient boosted tree
model_gbm <- gbm(Label ~ . , data = train_data_gbm, n.trees = 100, distribution = "bernoulli", interaction.depth = 9)

# Making predictions from test data
pred_gbm <- predict(model_gbm, test_data[-12])
pred_gbm <- factor(sapply(pred_gbm, function(x) ifelse(x >= 0, 1, 0)))

confusionMatrix(as.factor(pred_gbm), test_data$Label) # Plot confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 42 16
##          1  8 35
##
##                Accuracy : 0.7624
##                  95% CI : (0.6674, 0.8414)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : 1.001e-07
##
##                   Kappa : 0.5255
##
##  Mcnemar's Test P-Value : 0.153
##
##             Sensitivity : 0.8400
##             Specificity : 0.6863
##          Pos Pred Value : 0.7241
##          Neg Pred Value : 0.8140
##              Prevalence : 0.4950
##          Detection Rate : 0.4158
##    Detection Prevalence : 0.5743
##       Balanced Accuracy : 0.7631
##
##        'Positive' Class : 0
##
```

Tables to compare methods

```r
# function to calculate error rate
error_rate <- function(pred,test){
  counter <- 0
  for (i in 1:(nrow(test))){
    if(pred[i] != test[i,12])
       counter <- counter + 1
  }
  return(counter/nrow(test))
}
pred_rf <- model_rf$test$predicted
knitr::kable(matrix(nrow = 3, ncol = 2,
                    data = c("Decision tree",
                             "Gradient boosting",
                             "Random forest",
                             error_rate(pred_tree, test_data),
                             error_rate(pred_gbm, test_data),
                             error_rate(pred_rf, test_data))),
             col.names = c("Model type", "Error rate"),
             caption = "Error rates found for the different models tested.")
```

Error rates found for the different models tested.

Model type	Error rate
Decision tree	0.287128712871287
Gradient boosting	0.237623762376238
Random forest	0.227722772277228

```r
rf_tab <- as_tibble(colnames(train_data[,1:11])) %>%
  add_column(as.data.frame(model_rf$importance)) %>%
  rename(var = value)
gbm_tab <- as.data.frame(summary(model_gbm,plotit = F))  # Table with relative information of each independent variable

influence_tab <- inner_join(gbm_tab,  rf_tab, by = "var")
colnames(influence_tab) <- c("Variable", "Relative Influence", "Mean decerase gini")
knitr::kable(influence_tab, caption = "Table showing significance of each predictor in the model found through gradient boos
t and random forest")
```

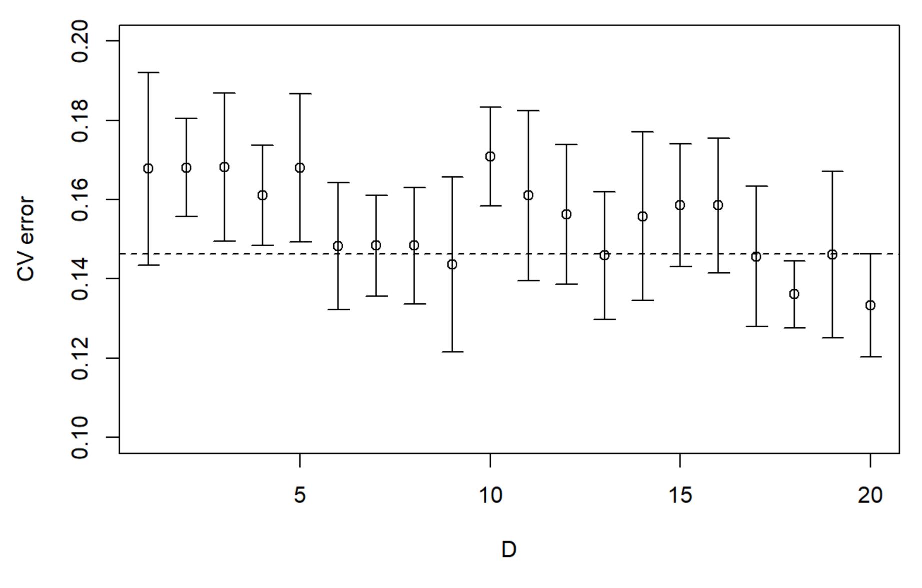
Table showing significance of each predictor in the model found through gradient boost and random forest

Variable	Relative Influence	Mean decerase gini
key	17.8436055	17.527328
speechiness	14.7040895	27.833312
danceability	14.3269159	26.680398
loudness	11.8052344	26.677345
energy	8.7736649	19.466566
tempo	8.0709349	16.069539
valence	7.4720522	18.842377
instrumentalness	7.1343359	15.000339
acousticness	4.7659638	20.601932
liveness	4.7199907	11.477311
mode	0.3832123	1.334406

Cross validation to find best parameters in gradient boosting

```r
# Function which returns the mean and standard deviation from 10-fold cross-validation
# given the input 'D', which is the interaction.depth used in gradient boosting

cv_Label_depth <- function(D){
  training_set <- train_data_gbm
  folds = createFolds(training_set$Label, k = 10)

  cv = lapply(folds, function(x) {
    training_fold = training_set[-x, ] # training fold =  training set minus (-) its sub test fold
    test_fold = training_set[x, ] # here we describe the test fold individually

    classifier = gbm(Label ~ . , data = training_fold,
                     n.trees = 100,
                     distribution = "bernoulli",
                     interaction.depth = D)

    Label_pred = as.data.frame(predict(classifier, newdata = test_fold[,-12]))
    Label_pred <- Label_pred %>%  mutate(pred = if_else(Label_pred[1] <= 0, 0 , 1))
    cm = table(as_vector(test_fold[, 12]), Label_pred$pred)
    accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
    return(accuracy)
  })

  cv_mean <- mean(as.numeric(cv))
  cv_sd <- sd(as.numeric(cv))
  return(c(cv_mean, cv_sd))
}

cv_Label_ntrees <- function(N){
  training_set <- train_data_gbm
  folds = createFolds(training_set$Label, k = 10)

  cv = lapply(folds, function(x) {
    training_fold = training_set[-x, ] # training fold =  training set minus (-) it's sub test fold
    test_fold = training_set[x, ] # here we describe the test fold individually

    classifier = gbm(Label ~ . , data = training_fold,
                     n.trees = N,
                     distribution = "bernoulli",
                     interaction.depth = 9) ## interaction depth 9 found in CV

    Label_pred = as.data.frame(predict(classifier, newdata = test_fold[,-12]))
    Label_pred <- Label_pred %>%  mutate(pred = if_else(Label_pred[1] <= 0, 0 , 1))
    cm = table(as_vector(test_fold[, 12]), Label_pred$pred)
    accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
    return(accuracy)
  })

  cv_mean <- mean(as.numeric(cv))
  cv_sd <- sd(as.numeric(cv))
  return(c(cv_mean, cv_sd))
}
```

```r
#Plotting mean cv-error against interaction.depth with error bars showing standard deviation
plot(D, 1-cv_means, ylab = "CV error", ylim = c(0.1,0.2))
arrows(D, 1-cv_means - cv_se, D, 1-cv_means + cv_se, code=3, angle=90, length=0.08)
abline(h = 1-cv_means[20]+cv_se[20], lty = 2)
```




```r
plot(N, 1-cv_means_ntree, ylab = "CV error", ylim = c(0.1,0.2))
arrows(N, 1-cv_means_ntree - cv_se_ntree, N, 1-cv_means_ntree + cv_se_ntree, code=3, angle=90, length=0.08)
abline(h = 1-cv_means_ntree[14]+cv_se_ntree[14], lty = 2)
```