Mathematical Methods for Artificial Intelligence Lab 4 - Random Forest and Boosting Alg.

Vytautas Kraujalis

2021-12-18

Contents

1 Random Forest **1**

1.1 Required packages 1

1.2 Parallel processing 2

1.3 Reading Data 2

1.4 Data Preparation 2

1.5 EDA, first look at the dataset 3

1.6 Fitting Random Forest Model 5

1.6.1 Tune with OOB 5

1.6.2 Random search 6

1.6.3 Grid Search 8

1.6.4 Tune number of trees 9

1.6.5 Optimal Random Forest model 10

1 Random Forest

1.1 Required packages

```
library(data.table)
library(SmartEDA)
library(dplyr)
library(ggplot2)
library(caret)
library(randomForest)
library(janitor)
library(tictoc)
library(tidyr)
library(tibble)
library(ggrepel)
```

```
# Function for 4 class accuracy from confusion matrix
classAcc <- function(confusionMatrix) {
  class1 <- round(confusionMatrix[1, 1] / sum(confusionMatrix[, 1]) * 100, 1)
  class2 <- round(confusionMatrix[2, 2] / sum(confusionMatrix[, 2]) * 100, 1)
  class3 <- round(confusionMatrix[3, 3] / sum(confusionMatrix[, 3]) * 100, 1)
  class4 <- round(confusionMatrix[4, 4] / sum(confusionMatrix[, 4]) * 100, 1)
  acc <- c(class1, class2, class3, class4 )
  names(acc) <- colnames(confusionMatrix)
  return(acc)
}
```

1.2 Parallel processing

```
library(parallel)
no_cores <- detectCores() - 1
library(doParallel)
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)
```

1.3 Reading Data

```
set.seed(123)

data_original <- fread("activity.csv")
data_names <- read.table("names.txt") %>%
   rename(column_names = V1)
```

1.4 Data Preparation

```
length(data_names$column_names)
```

```
## [1] 535
```

```
n_distinct(data_names$column_names)
```

```
## [1] 417
```

There are 535 provided column names, but only 417 are distinct, it means we have some duplicated names, we need to make them unique. To do that, for duplicated names we'll add a unique ID at the end:

```
data <- data_original
colnames(data) <- data_names$column_names

data <- data %>%
   clean_names()

n_distinct(colnames(data))
```

```
## [1] 535
```

1.5 EDA, first look at the dataset

```
ExpData(data,type = 1)
```

```
##                                          Descriptions      Value
## 1                                  Sample size (nrow)       4480
## 2                             No. of variables (ncol)        535
## 3                     No. of numeric/interger variables        534
## 4                               No. of factor variables          0
## 5                                 No. of text variables          1
## 6                              No. of logical variables          0
## 7                           No. of identifier variables          0
## 8                                 No. of date variables          0
## 9              No. of zero variance variables (uniform)          4
## 10                %. of variables having complete cases 100% (535)
## 11   %. of variables having >0% and <50% missing cases     0% (0)
## 12 %. of variables having >=50% and <90% missing cases     0% (0)
## 13          %. of variables having >=90% missing cases     0% (0)
```

We have a dataset of 4480 observations with 535 variables, only 1 variable has text format. All variables have no missing values. We can see that there are 4 variables with zero variance, we'll remove those later.

Let's look at the response variable:

```
data %>%
   group_by(activity) %>%
   summarise(n = n()) %>%
   mutate(n_prop = round(n / sum(n) * 100, 2))
```

```
## # A tibble: 4 x 3
##   activity      n n_prop
##   <chr>     <int>  <dbl>
## 1 emotional  1120     25
## 2 mental     1120     25
## 3 neural     1120     25
## 4 physical   1120     25
```

We have perfectly balanced response variable with 4 classes.

We'll change the response variable to factor type.

```
data <- data %>%
   mutate_if(is.character, as.factor)
```

```
data %>%
   select(nearZeroVar(data)) %>%
   summary()
```

```
##  ecg_p_vfl_kurtosis ecg_p_lf_kurtosis it_vlf_kurtosis it_lf_kurtosis_2
##  Min.   :1.5        Min.   :1         Min.   :1.5     Min.   :1
##  1st Qu.:1.5        1st Qu.:1         1st Qu.:1.5     1st Qu.:1
##  Median :1.5        Median :1         Median :1.5     Median :1
##  Mean   :1.5        Mean   :1         Mean   :1.5     Mean   :1
##  3rd Qu.:1.5        3rd Qu.:1         3rd Qu.:1.5     3rd Qu.:1
##  Max.   :1.5        Max.   :1         Max.   :1.5     Max.   :1
```

As mentioned previously, we have 4 variables with zero variance, we will remove those columns.

```
data <- data %>%
   select(-nearZeroVar(data))
```

We should look at the correlation between variables

```
# Correlation

corr_simple <- function(df,sig=0.5){
  corr <- cor(df)
  #prepare to drop duplicates and correlations of 1
  corr[lower.tri(corr,diag=TRUE)] <- NA
  #drop perfect correlations
  corr[corr == 1] <- NA
  #turn into a 3-column table
  corr <- as.data.frame(as.table(corr))
  #remove the NA values from above
  corr <- na.omit(corr)
  #select significant values
  corr <- subset(corr, abs(Freq) > sig)
  #sort by highest correlation
  corr <- corr[order(-abs(corr$Freq)),]
  return(corr)
}

correlation_matrix = cor(data %>% select(-activity, -subject_index))

length(findCorrelation(correlation_matrix, cutoff = 0.99))
```

```
## [1] 309
```

```
length(findCorrelation(correlation_matrix, cutoff = 0.95))
```

```
## [1] 369
```

```
length(findCorrelation(correlation_matrix, cutoff = 0.9))
```

```
## [1] 389
```

We have 309 variables with correlation greater than 0.99, we will remove those variables.

```
data <- data %>%
   select(-findCorrelation(correlation_matrix, cutoff = 0.99))
```

We'll convert subject index column to factor type:

```
data %>%
   select(subject_index) %>%
   table()
```

```
## .
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
## 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112 112
```

```
data <- data %>%
   mutate(subject_index = as.factor(subject_index))
```

Let's look how obersvations are distributed across subjects and activity types:

```
data %>%
   group_by(subject_index, activity) %>%
   summarise(n = n()) %>%
   pivot_wider(names_from = activity, values_from = n)
```

```
## # A tibble: 40 x 5
## # Groups:   subject_index [40]
##    subject_index emotional mental neural physical
##    <fct>             <int>  <int>  <int>    <int>
##  1 1                    28     28     28       28
##  2 2                    28     28     28       28
##  3 3                    28     28     28       28
##  4 4                    28     28     28       28
##  5 5                    28     28     28       28
##  6 6                    28     28     28       28
##  7 7                    28     28     28       28
##  8 8                    28     28     28       28
##  9 9                    28     28     28       28
## 10 10                   28     28     28       28
## # ... with 30 more rows
```

Each person has 28 observations of each activity. We are not going to use Out-of-Bag score for tuning parameters as this would be misleading. Same person with the same activity could be splited into different sets and the result based on OOB could be misleading. We are not going to have this problem with Cross Validation, as we can specify folds to be grouped according to subjects.

1.6 Fitting Random Forest Model

1.6.1 Tune with OOB

Let's look at the optimal mtry value based on OOB and number of trees = 500

```
tic()
png(file = "Tune_oob.png", width = 1200, height = 850)
rf_tune <- tuneRF(data %>%
                     select(-subject_index, -activity),
                  data$activity,
                  mtryStart = 2,
                  ntreeTry = 500,
                  stepFactor = 3,
                  improve = 0.001,
                  trace = TRUE,
                  plot = TRUE)
```

```
## mtry = 2  OOB error = 1.27%
## Searching left ...
## mtry = 1     OOB error = 2.48%
## -0.9473684 0.001
## Searching right ...
## mtry = 6     OOB error = 1.38%
## -0.0877193 0.001
```

```
dev.off()
```

```
## pdf
##   2
```

```
toc()
```

```
## 53.91 sec elapsed
```

Based on OOB the optimal mtry was found to be 2. We are not going to use this value as mentioned previously.

1.6.2 Random search

```
folds = groupKFold(data$subject_index, k = 10)
fitControl <- trainControl(## 10-fold CV
  index = folds,
  method = "cv",
  number = 10,
  classProbs = TRUE,
  savePredictions='all',
  verboseIter = TRUE,
  allowParallel = TRUE,
  search = "random")

tic()
rf_random <- train(
   activity ~ .,
   data = data %>% select(-subject_index),
```

```
   method = "rf",
   metric = "Accuracy",
   tuneLength = 10,
   trControl = fitControl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 2 on full training set
```

```
toc()
```

```
## 352.05 sec elapsed
```

```
print(rf_random)
```

```
## Random Forest
##
## 4480 samples
##  220 predictor
##    4 classes: 'emotional', 'mental', 'neural', 'physical'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4256, 4256, 4032, 3920, 3584, 4368, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.7234375  0.6312500
##    52   0.7200446  0.6267262
##    61   0.7193080  0.6257440
##    67   0.7191518  0.6255357
##    75   0.7177009  0.6236012
##    82   0.7200000  0.6266667
##   101   0.7155357  0.6207143
##   118   0.7154911  0.6206548
##   164   0.7118304  0.6157738
##   188   0.7123437  0.6164583
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
png(file = "Tune_randomSearch.png", width = 1200, height = 850)
plot(rf_random)
dev.off()
```

```
## pdf
##   2
```

Based on random search, the optimal mtry was found to be 2. We are going to also use a grid search, to look if an optimal mtry could be smalle and closer to sqrt(m) value.

1.6.3 Grid Search

```
folds = groupKFold(data$subject_index, k = 10)
fitControl <- trainControl(## 10-fold CV
  index = folds,
  method = "cv",
  number = 10,
  classProbs = TRUE,
  savePredictions='all',
  verboseIter = TRUE,
  allowParallel = TRUE,
  search = "grid")

rf_grid = expand.grid(.mtry = c(seq(2,15,2), seq(rf_random$bestTune$mtry - 4, rf_random$bestTune$mtry +

tic()
rf_fit_grid_cv <- train(
   activity ~ .,
   data = data %>% select(-subject_index),
   method = "rf",
   metric = "Accuracy",
   tuneGrid = rf_grid,
   trControl = fitControl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 10 on full training set
```

```
toc()
```

```
## 192.99 sec elapsed
```

```
print(rf_fit_grid_cv)
```

```
## Random Forest
##
## 4480 samples
##  220 predictor
##    4 classes: 'emotional', 'mental', 'neural', 'physical'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4144, 3584, 4032, 4144, 4032, 3584, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   -2    0.7031085  0.6041446
##    0    0.6996776  0.5995701
##    2    0.7071181  0.6094907
##    4    0.7101438  0.6135251
##    6    0.7241237  0.6321649
```

```
##    8    0.7255622  0.6340829
##   10    0.7281415  0.6375220
##   12    0.7229332  0.6305776
##   14    0.7230985  0.6307981
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.
```

```
png(file = "Tune_gridSearch.png", width = 1200, height = 850)
plot(rf_fit_grid_cv)
dev.off()
```

```
## pdf
##   2
```

Using grid search we found an optimal value of mtry to be 10. We are going to use this value in our further analysis.

1.6.4 Tune number of trees

```
folds = groupKFold(data$subject_index, k = 10)
fitControl <- trainControl(## 10-fold CV
  index = folds,
  method = "cv",
  number = 10,
  classProbs = TRUE,
  savePredictions='all',
  verboseIter = TRUE,
  allowParallel = TRUE,
  search = "grid")

rf_grid = expand.grid(.mtry = rf_fit_grid_cv$bestTune$mtry)

modellist <- list()
for (ntree in c(350, 500, 750, 1000, 1500)) {
    fit <- train(
   activity ~ .,
   data = data %>% select(-subject_index),
   method = "rf",
   metric = "Accuracy",
   tuneGrid = rf_grid,
   trControl = fitControl,
   ntree = ntree)
    key <- toString(ntree)
    modellist[[key]] <- fit
}
```

```
## Aggregating results
## Fitting final model on full training set
## Aggregating results
## Fitting final model on full training set
```

```
## Aggregating results
## Fitting final model on full training set
## Aggregating results
## Fitting final model on full training set
## Aggregating results
## Fitting final model on full training set
```

```
# compare results
results <- resamples(modellist)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: 350, 500, 750, 1000, 1500
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 350  0.5848214 0.6928013 0.7444196 0.7280432 0.7814732 0.8616071    0
## 500  0.6160714 0.6994978 0.7493304 0.7355134 0.7836310 0.8616071    0
## 750  0.5848214 0.6964286 0.7491071 0.7307440 0.7853423 0.8616071    0
## 1000 0.5982143 0.6947545 0.7419643 0.7286756 0.7779018 0.8392857    0
## 1500 0.5892857 0.6944754 0.7431548 0.7275074 0.7681920 0.8571429    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 350  0.4464286 0.5904018 0.6592262 0.6373909 0.7086310 0.8154762    0
## 500  0.4880952 0.5993304 0.6657738 0.6473512 0.7115079 0.8154762    0
## 750  0.4464286 0.5952381 0.6654762 0.6409921 0.7137897 0.8154762    0
## 1000 0.4642857 0.5930060 0.6559524 0.6382341 0.7038690 0.7857143    0
## 1500 0.4523810 0.5926339 0.6575397 0.6366766 0.6909226 0.8095238    0
```

```
png(file = "Tune_ntree.png", width = 1200, height = 850)
dotplot(results)
dev.off()
```

```
## pdf
##   2
```

```
mtry_optimal <- rf_fit_grid_cv$bestTune$mtry

ntree_optimal <- as.integer(names(which(summary(results)$statistics$Accuracy[,"Mean"] == max(summary(res
```

Optimal number of trees were found to be 500.

1.6.5 Optimal Random Forest model

Fit a random forest model with best parameters (CV)

```
folds = groupKFold(data$subject_index, k = 10)
fitControl <- trainControl(## 10-fold CV
  index = folds,
  method = "cv",
  number = 10,
  classProbs = TRUE,
  savePredictions='all',
  verboseIter = TRUE,
  allowParallel = TRUE)

tic()
rf_fit_cv <- train(
   activity ~ .,
   data = data %>% select(-subject_index),
   method = "rf",
   tuneGrid = expand.grid(.mtry = mtry_optimal),
   ntree = ntree_optimal,
   trControl = fitControl,
   importance = TRUE)
```

```
## Aggregating results
## Fitting final model on full training set
```

```
toc()
```

```
## 95.77 sec elapsed
```

```
rf_final <- rf_fit_cv$finalModel
```

```
print(rf_final)
```

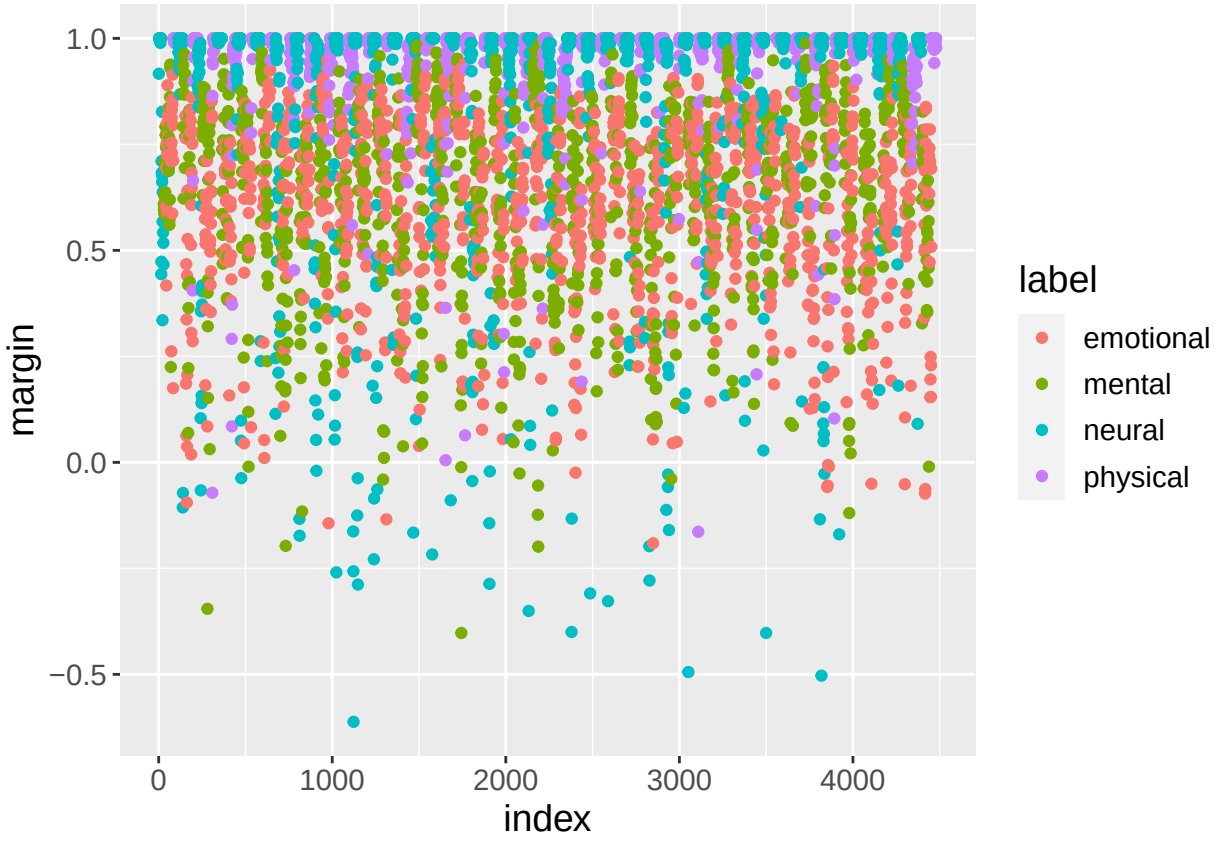
```
##
## Call:
##  randomForest(x = x, y = y, ntree = ..1, mtry = min(param$mtry,      ncol(x)), importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 1.56%
## Confusion matrix:
##           emotional mental neural physical class.error
## emotional      1107     13      0        0 0.011607143
## mental           13   1106      1        0 0.012500000
## neural           22     19   1079        0 0.036607143
## physical          1      1      0     1118 0.001785714
```

```
confusion_matrix <- rf_final$confusion[,1:4]
classAcc(confusion_matrix)
```

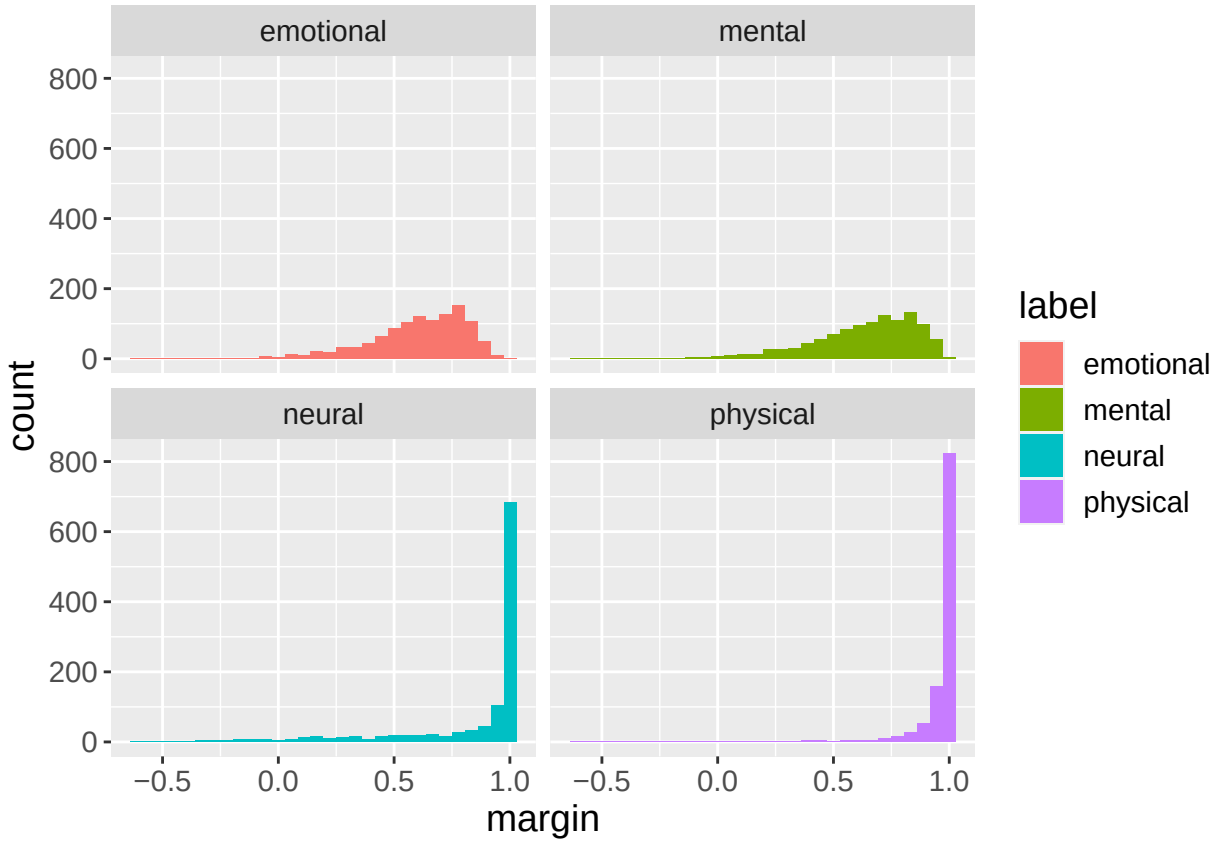
```
## emotional    mental    neural  physical
##      96.9      97.1      99.9     100.0
```

Final model has accuracy of 98.44%.

```
rf_margin <- margin(rf_final, data$activity)
df = data.frame(margin = as.numeric(rf_margin),
                label = names(rf_margin),
                index =  1:length(rf_margin))
df %>%
   ggplot(aes(x = index,y = margin,col = label)) +
   geom_point() +
   theme(text = element_text(size = 14))
```




```
ggsave(filename = "Margin_ScatterPlot.png", width = 14, height = 7, units = "in", bg = "white")

df %>%
   ggplot(aes(x = margin, fill = label)) +
   geom_histogram() +
   facet_wrap(~label) +
   theme(text = element_text(size = 14))
```



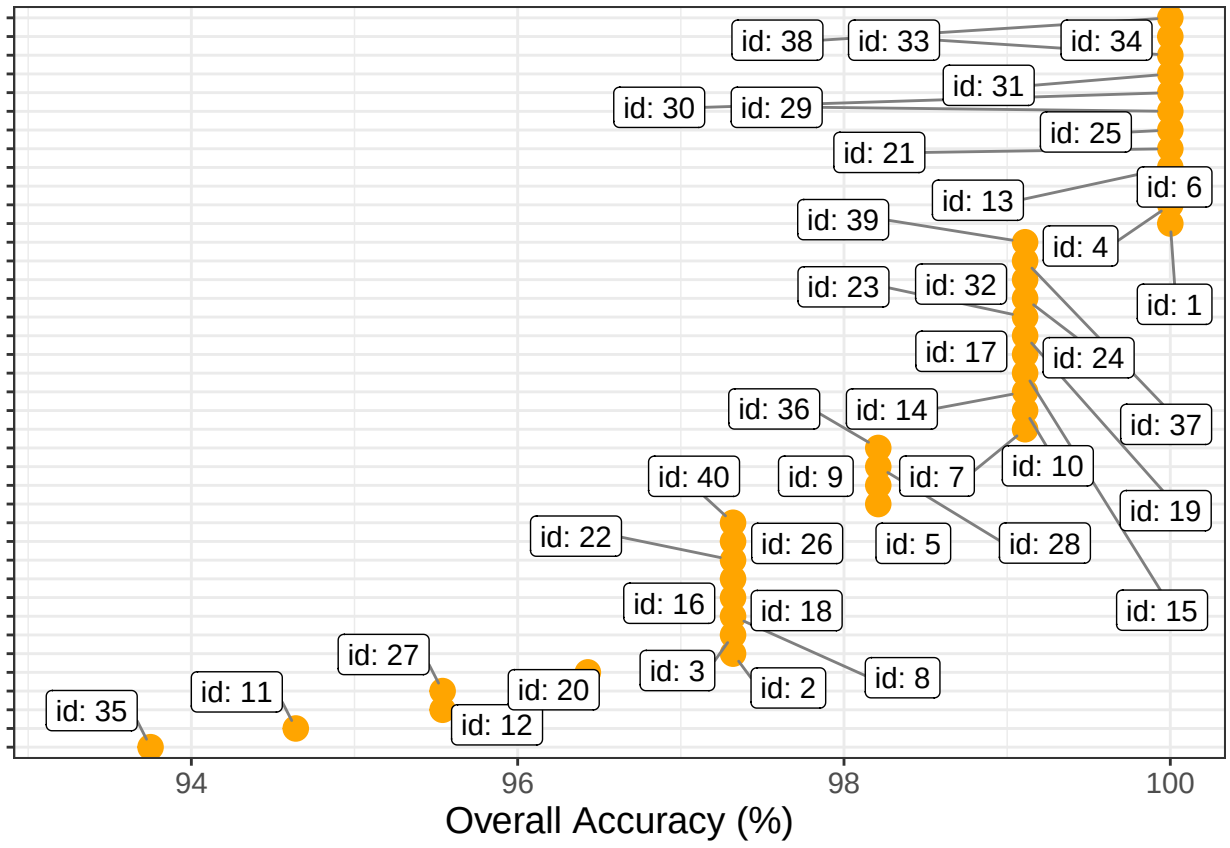

```
ggsave(filename = "Margin_BarPlot.png", width = 14, height = 7, units = "in", bg = "white")
```

We clearly see, that only a small number of observations were misclassified (margin < 0). We also clearly see that majority of Neutral and Physical activity observations were classified correctly with almost perfect voting score for the correct class. One could also identify, that for classes Emotional and Mental, most of the correct guesses were made with ~0.5 - ~0.75 majority votes. To increase the model effectiveness, one should find more features to distinguish Emotional and Mental classes.

Let's explore model performance for each subject:

```
# get a table of actual and predicted values for each subject
df <- data.frame(
   subject_id = data$subject_index,
   activity = rf_final$y,
   activity_prediction = rf_final$predicted
)

# list of tables by each subject
df_subject <- split(df, df$subject_id)

# list of confusion matrices by each subject
df_subject <- lapply(df_subject, function(x){confusionMatrix(x$activity_prediction, x$activity)$table})

# list of class and overall accuracies by each subject
df_subject <-lapply(df_subject, function(x){
   AccClasses = classAcc(x)
```

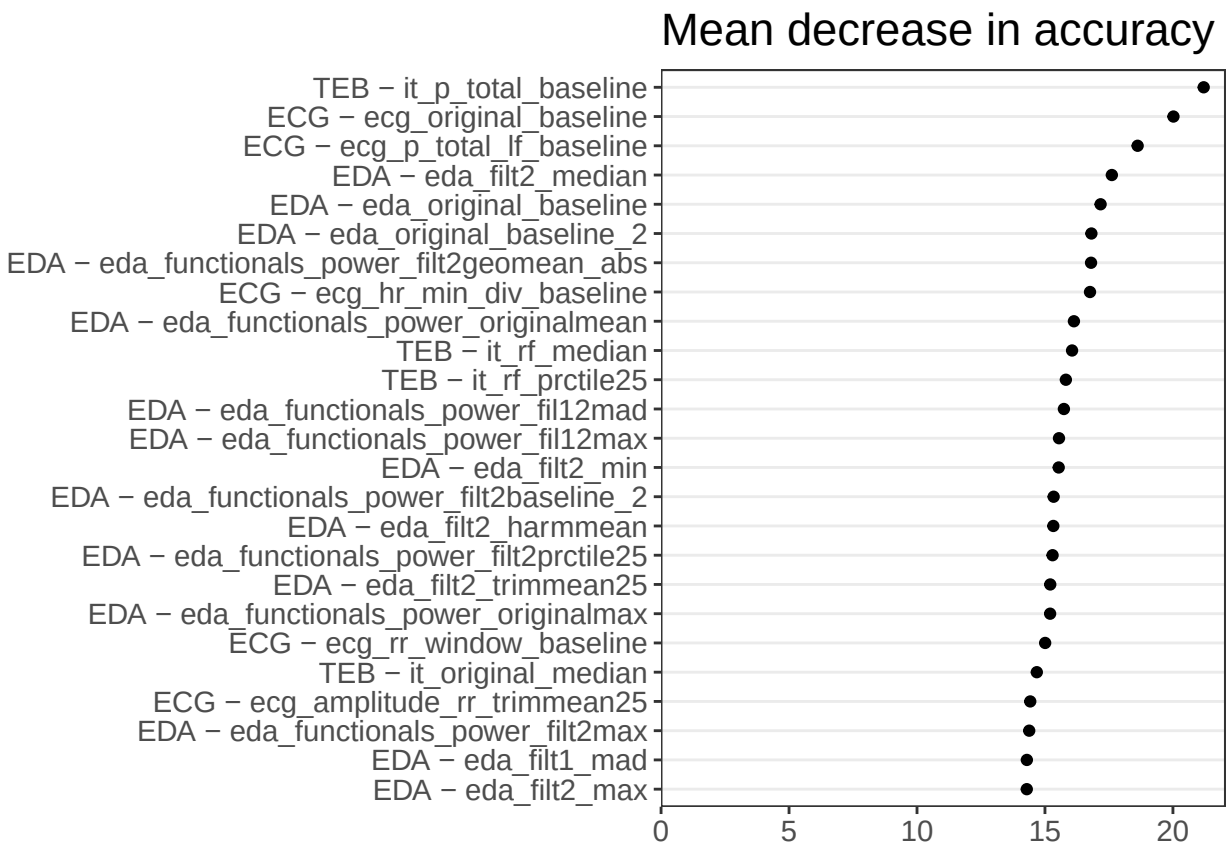
```
   AccOverall = round(sum(diag(x)) / sum(colSums(x)) * 100, 2)
   acc = c(AccClasses, AccOverall)
   names(acc) = c(names(AccClasses), "OVERALL")
   return(acc)
})

df_colnames <- names(df_subject[[1]])

# a table of class and overall accuracies by each subject
df_subject <- do.call(rbind.data.frame, df_subject) %>%
   rownames_to_column("subject_id") %>%
   mutate(subject_id = as.numeric(subject_id))
colnames(df_subject) <- c("subject_id", df_colnames)

# Plot of overall
df_subject %>%
  arrange(OVERALL) %>%
  mutate(subject_id = factor(subject_id, levels = subject_id)) %>%
  ggplot( aes(x = subject_id, y = OVERALL)) +
   geom_point( size = 4, color = "orange") +
   ylab("Overall Accuracy (%)") +
   coord_flip() +
   theme_bw() +
   lims(y = c(min(df_subject$OVERALL) - 0.5, 100)) +
   geom_label_repel(aes(label = paste0("id: ", subject_id)),
                    box.padding   = 0.35,
                    point.padding = 0.5,
                    segment.color = 'grey50') +
   theme(axis.text.y=element_blank(),
         axis.title.y=element_blank(),
         text = element_text(size = 14))
```




```
ggsave(filename = "OverallAccBySubject.png", width = 14, height = 7, units = "in", bg = "white")

df_subject %>%
   pivot_longer(cols = c(-subject_id), names_to = "Type", values_to = "Accuracy") %>%
   group_by(Type, Accuracy_Rounded = round(Accuracy)) %>%
   summarise(n = n()) %>%
   pivot_wider(names_from = Type, values_from = n, values_fill = 0) %>%
   arrange(desc(Accuracy_Rounded))
```

```
## # A tibble: 11 x 6
##    Accuracy_Rounded emotional mental neural OVERALL physical
##               <dbl>     <int>  <int>  <int>   <int>    <int>
##  1              100        31     29     17      12       38
##  2               99         0      0      0      11        0
##  3               98         0      0      0       4        0
##  4               97         0      0      0       8        0
##  5               96         7      9     14       3        2
##  6               95         0      0      0       1        0
##  7               94         0      0      0       1        0
##  8               93         1      1      4       0        0
##  9               89         0      1      3       0        0
## 10               86         1      0      1       0        0
## 11               79         0      0      1       0        0
```

```r
df <- importance(rf_final) %>%
   as.data.frame() %>%
   rownames_to_column("feature") %>%
   mutate(
      feature_type = case_when(
         substr(feature, 0, 2) == "ec" ~ "ECG",
         substr(feature, 0, 2) == "it" ~ "TEB",
         substr(feature, 0, 2) == "ed" ~ "EDA",
         TRUE ~ "ERROR"
      )
   ) %>%
   mutate(feature = paste0(feature_type, " - ", feature))

df %>%
   arrange(MeanDecreaseAccuracy) %>%
   tail(25) %>%
   mutate(feature = factor(feature, levels = feature)) %>%
   ggplot(aes(MeanDecreaseAccuracy, feature)) +
   geom_point() +
   scale_x_continuous(limits=c(0,NA), expand=expansion(c(0,0.04))) +
   theme_bw() +
   theme(panel.grid.minor=element_blank(),
         panel.grid.major.x=element_blank(),
         panel.grid.major.y=element_line(),
         axis.title=element_blank(),
         text = element_text(size = 14)) +
   labs(title = "Mean decrease in accuracy")
```




```
ggsave(filename = "VariableImportance.png", width = 14, height = 7, units = "in", bg = "white")

temp <- df %>%
   arrange(MeanDecreaseAccuracy) %>%
   group_by(feature_type) %>%
   summarise(n_all = n())

df %>%
   arrange(MeanDecreaseAccuracy) %>%
   tail(25) %>%
   group_by(feature_type) %>%
   summarise(n = n()) %>%
   left_join(temp, by = "feature_type") %>%
   mutate(n_prop = n / n_all * 100)
```

```
## # A tibble: 3 x 4
##   feature_type     n n_all n_prop
##   <chr>        <int> <int>  <dbl>
## 1 ECG              5    91   5.49
## 2 EDA             16    68  23.5
## 3 TEB              4    61   6.56
```