FeatureSelection_LDA_MARS

Milton Osiel Candela Leal

7/17/2021

Packages used

The following packages were used:

```
library(R.matlab)
library(dplyr)
library(caret)
library(MASS)
library(earth)
```

Getting the Data Frame

Similar to the previous milestone, the *WFM.1* document is extracted from the .mat document, change the columns' names and filter the dataset making sure all values on the Data Frame are less than 20 (due to the fact that the data is already standardized by its mean, it is really strange to have values superior to 20). The features from the Empatica E4 wristband are further eliminated, although there is class imbalance problem among the predicted classes.

```
data <- readMat('Data/Raw/FeatureMatrices.mat')
unmin <- as.data.frame(data$WFM.1)

feature_names <- unlist(data$FMinfo)
feature_names <- gsub(' \\(', '_', feature_names)
feature_names <- gsub(')', '', feature_names)
feature_names <- gsub('`', '', feature_names)
feature_names <- gsub('\\/', '-', feature_names)

colnames(unmin) <- feature_names

df <- filter_all(unmin, all_vars(. < 20))
df <- mutate(df, Clas = as.factor(Clas))

EDA_index <- 121:129
div_index <- 41:120

df <- df[,-EDA_index]

print(summary(df$Clas))
```

```
##   1   2   3
##  69 483 205
```

This imbalance of classes was solved via an up-sampling technique given by the caret package, and so all the classes have the same number of samples.

```
set.seed(2031)
df <- upSample(df[,-length(df)], df$Clas, yname = 'Clas')
print(summary(df$Clas))
```

```
##   1   2   3
## 483 483 483
```

Feature selection using RFE and MARS

An hybrid feature selection technique was used in order to reduce the number of features used, as the previous generated models used all of the features available.

The first feature selection method us called Recursive Feature Elimination (RFE), using Linear Discriminant Analysis (LDA) to reduce the number of features to a desired value (in this case, the number 44 was employed). This provived the top 44 features according to the method used, then, the data was fitted into a Multivariate Regression Spinelines (MARS) model to capture non-linearity between features, which further reduced the number of features to 10.

```
ldaProf <- function(df, sizes = 1:50) {
     set.seed(1002)

     inTrain <- createDataPartition(df$Clas, p = 0.70, list = FALSE)[,1]

     train <- df[inTrain, -length(df)]
     trainClass <- df$Clas[inTrain]

     set.seed(302)
     ldaProfile <- rfe(train, trainClass, sizes = sizes,
                       rfeControl = rfeControl(functions = ldaFuncs, method = 'cv'))
     return(ldaProfile)
}

dfTopN <- function(df, n = 44){
     topn <- head(ldaProf(df)$variables$var, n)
     df <- df[,c(topn, 'Clas')]

     marsModel <- earth(Clas ~ ., data = df, pmethod = 'cv', nfold = 5)
     ev <- evimp(marsModel)

     nombresMARS <- gsub("`", "", rownames(ev))
     df <- df[,c(nombresMARS, 'Clas')]

     return(df)
}

df <- dfTopN(df, n = 44)
```

With this filtered Data Frame, the top features are the following (excluding *Clas*):

```r
print(colnames(df))
```

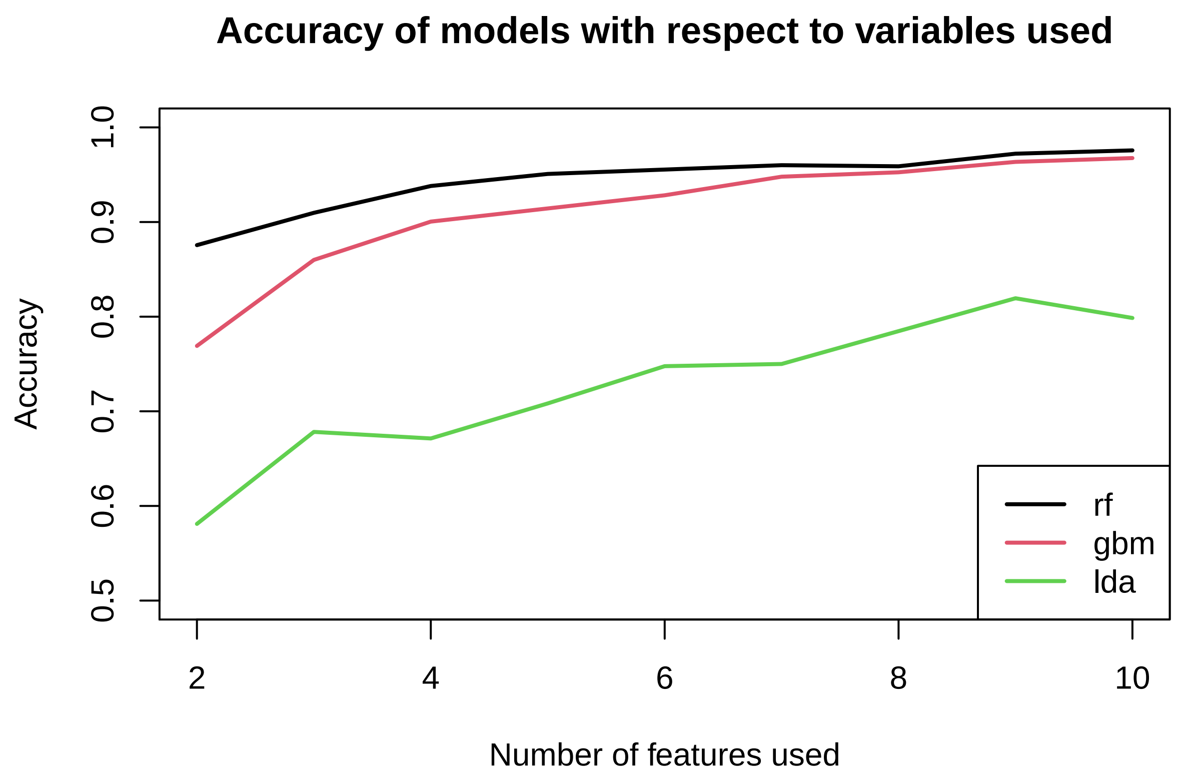
```
##  [1] "Gamma_01"   "t-g_FP1"    "Beta_FP1"   "t-g_P7"     "Theta_P7"   "Delta_FP2"
##  [7] "Gamma_P7"   "a-b_01"     "d-g_P8"     "Delta_P8"   "Clas"
```

Generalized testing using gbm, rf, lda

In order to test this features' performance, a generalized testing function was created, which separates the Data Frame into training and testing dataset, fits an algorithm, averages the accuracy score using a confusion matrix and incremental cross-validations from 1 to 5. Then, a for loop incremented the number of features from 2 to 10, in order to appreciate the correct number of features required by the algorithms used.

```r
geneTesting <- function(df){
     set.seed(1002)

     trainIndex <- createDataPartition(df$Clas, p = 0.70, list = FALSE)
     training <- df[trainIndex,]
     testing <- df[-trainIndex,]

     prob_modelo <- function(training, met, testing, control){

          if(met == 'gbm'){
               modelo <- train(Clas ~ ., data = training, method = met, trControl = control,
                               verbose = FALSE)
          } else if (met == 'rf'){
               modelo <- train(Clas ~ ., data = training, method = met, trControl = control,
                               tuneLength = length(train))
          } else {
               modelo <- train(Clas ~ ., data = training, method = met, trControl = control)
          }

          Pred <- predict(modelo, testing)
          acc <- round(confusionMatrix(testing$Clas, Pred)$overall[1], 5)
          return(acc)
     }

     n <- 5
     metodos <- c('rf', 'gbm', 'lda')
     df_modelos <- data.frame(rf = NA, gbm = NA, lda = NA)
     for(metodo in metodos){
          accs <- c()
          for(num in 2:n){
               control <- trainControl(method = 'cv', number = num)
               acc <- prob_modelo(training, metodo, testing, control)
               accs <- c(accs, acc)
          }
          df_modelos[,metodo] <- mean(accs)
     }
     return(df_modelos)
}
```

```r
n <- length(df) - 1
dfEval <- data.frame(n_vars = 2:n, rf = NA, gbm = NA, lda = NA)

for(i in 2:n){
     dfEval[dfEval$n_vars == i,c('rf', 'gbm', 'lda')] <- geneTesting(df[,c(1:i, length(df))])
}

plot(dfEval$rf ~ dfEval$n_vars, col = 1, lwd = 2, type = 'l', xlab = 'Number of features used', ylab =
     main = 'Accuracy of models with respect to variables used', ylim = c(0.5, 1), xlim = c(2,n))
lines(dfEval$gbm ~ dfEval$n_vars, col = 2, lwd = 2)
lines(dfEval$lda ~ dfEval$n_vars, col = 3, lwd = 2)
legend('bottomright', legend = c('rf', 'gbm', 'lda'), col = 1:3, lty = 1, lwd = 2)
```



As it can be seen, with default parameters, Random Forest (rf) outperforms Gradient Boosting Method (gbm) and Linear discriminant analysis (lda) on a small set of features. However, as the number of features increased, the performance of gbm is compared to rf, while lda's performance is somewhat lower than the performance of the aforementioned algorithms.