Evaluation_Documents

Milton Candela

7/29/2021

Packages used

The following packages were used:

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

Class imbalance reduction & boxplots

Both WFM.1 and WFM.4 documents will be analyzed in order to detect the differences between these documents. The *.mat* files were used to create *.csv* files, and so to directly get a DataFrame in R via the function *read.csv*. Two *.csv* files were created based on the document and its most important columns, these files are named *featureBoth1.csv* for the WFM.1 document and *featureBoth4.csv* for the WFM.4 document.

It is worth noting that the same feature selection method (viewed on the previous milestones) was applied on both documents, although, the WFM.4 document had more important features and could not be reduced to ten columns when applying the last MARS method (it had 16 features instead of 10). However, in order to make the datasets equal, the top 10 features were extracted from the MARS model on the *WFM.4* data, but this may indicate that the data found on the *WFM.4* document is more complex and so it needed more features to explain all its variability.

The script created in order to create these *.csv* files can be found on the same milestone's folder, with the name *combPreFeat.R*. As it not only uses *FeatureMatrices.mat*, but also *FeatureMatrices_pre.mat*, that is, it combines the data found previous to the test, with the data collected during the test. The script also adds a filter of -10 and 10, so that all the values would have a domain that is less than 10 but more than -10.

WFM.1's analysis

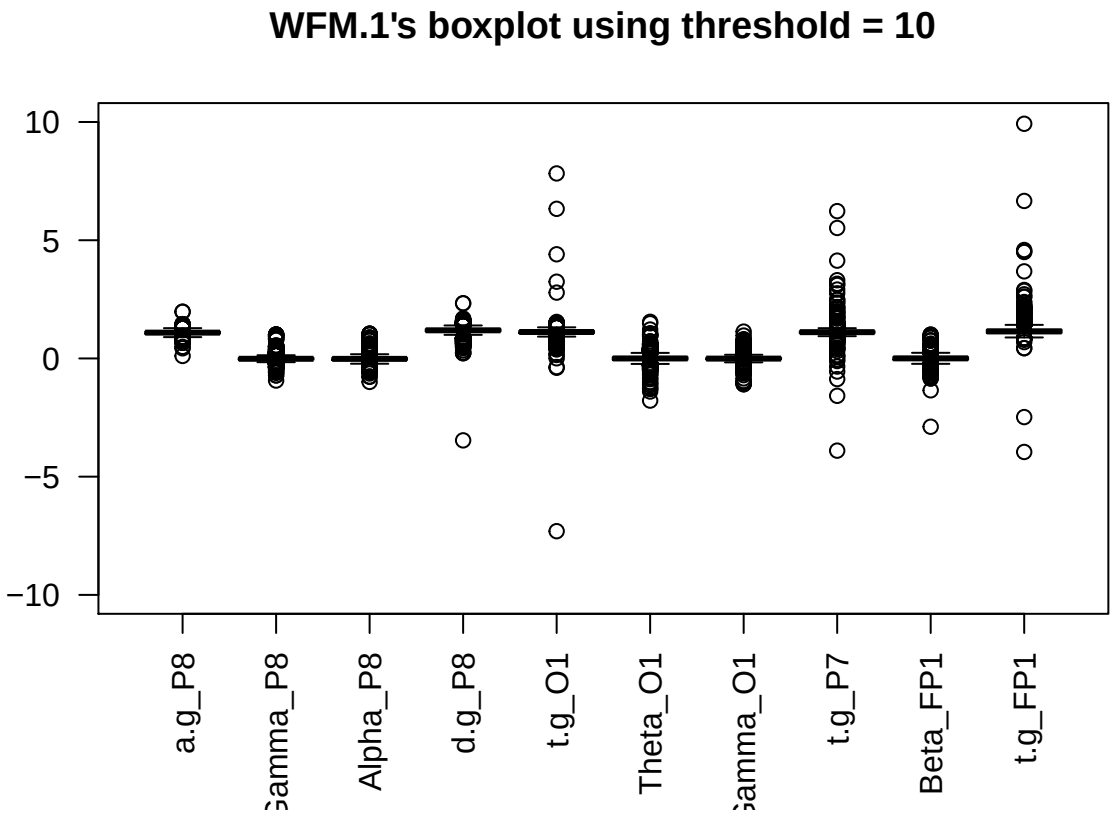
```
featureBoth1 <- read.csv('Data/Processed/featureBoth1.csv')
featureBoth1 <- dplyr::mutate(featureBoth1, Clas = as.factor(Clas))
writeLines("WFM.1's class distribution using random Up-Sampling")
```

```
## WFM.1's class distribution using random Up-Sampling
```

```
print(summary(featureBoth1$Clas))
```

```
##    1    2    3
## 1349 1349 1349
```

```
boxplot(featureBoth1[,-length(featureBoth1)], las = 2, ylim = c(-10,10),
        main = "WFM.1's boxplot using threshold = 10")
```



The class imbalance problem is solved with a random Up-Sampling method, and so accuracy can be used to asses the effectiveness of Machine Learning models' predictions. Moreover, as it can be seen on the box plot, the majority of the data is concentrated between -2.5 and 2.5, with some outliers that has values superior than 5 and inferior than -5, although, the data seems to be well normalized and with little to no noise.

WFM.4's analysis

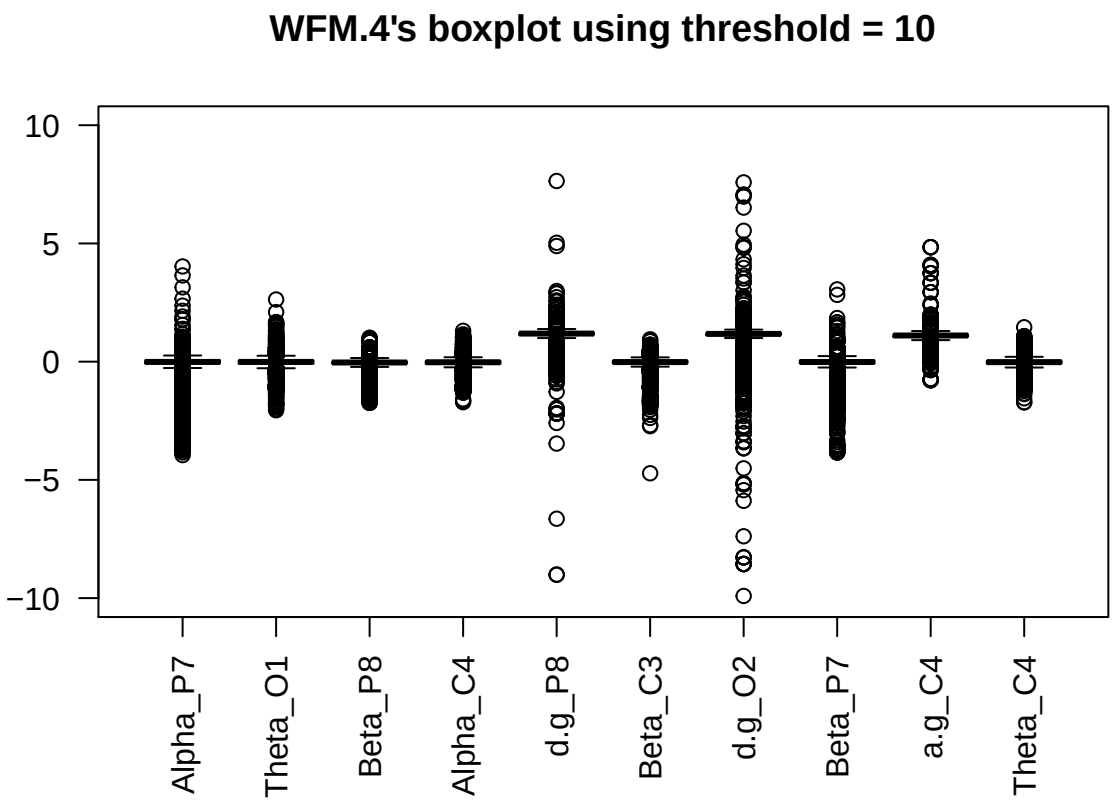
```
featureBoth4 <- read.csv('Data/Processed/featureBoth4.csv')
featureBoth4 <- dplyr::mutate(featureBoth4, Clas = as.factor(Clas))
writeLines("WFM.4's class distribution using random Up-Sampling")
```

```
## WFM.4's class distribution using random Up-Sampling
```

```
print(summary(featureBoth4$Clas))
```

```
##    1    2    3
## 5389 5389 5389
```

```r
boxplot(featureBoth4[,-length(featureBoth4)], las = 2, ylim = c(-10,10),
        main = "WFM.4's boxplot using threshold = 10")
```



Similar to the WFM.1 dataset, the *.csv* file already has the Up-Sampling method implemented, as there is no majority of classes. Furthermore, the box plot seems to be kind of different; even though the features do not match, the data is supposedly normalized with their mean, and so the distribution should be pretty equal. Although, it seems to be that there is more variability in this dataset, as most of the features had values between -5 and 5, while some ratio features have some outliers that surpass that threshold.

Model creation

Two Random Forest (rf) models (named *model1* and *model4*) would be created using both the WFM.1 and WFM.4 dataset, as rf was superior than gbm (proved on the previous milestone report). This model uses 5-fold cross-validations, 50 trees to reduce overfitting, the same 70:30 partition between training and testing, and its restricted to the top 10 features.

```r
getModel <- function(df, n = 10){
     df <- df[,c(1:n, length(df))]

     set.seed(1002)
     trainIndex <- createDataPartition(df$Clas, p = 0.70, list = FALSE)
     training <- df[trainIndex,]

     control <- trainControl(method = 'cv', number = 5)
     modelo <- train(Clas ~ ., data = training, method = 'rf',
```

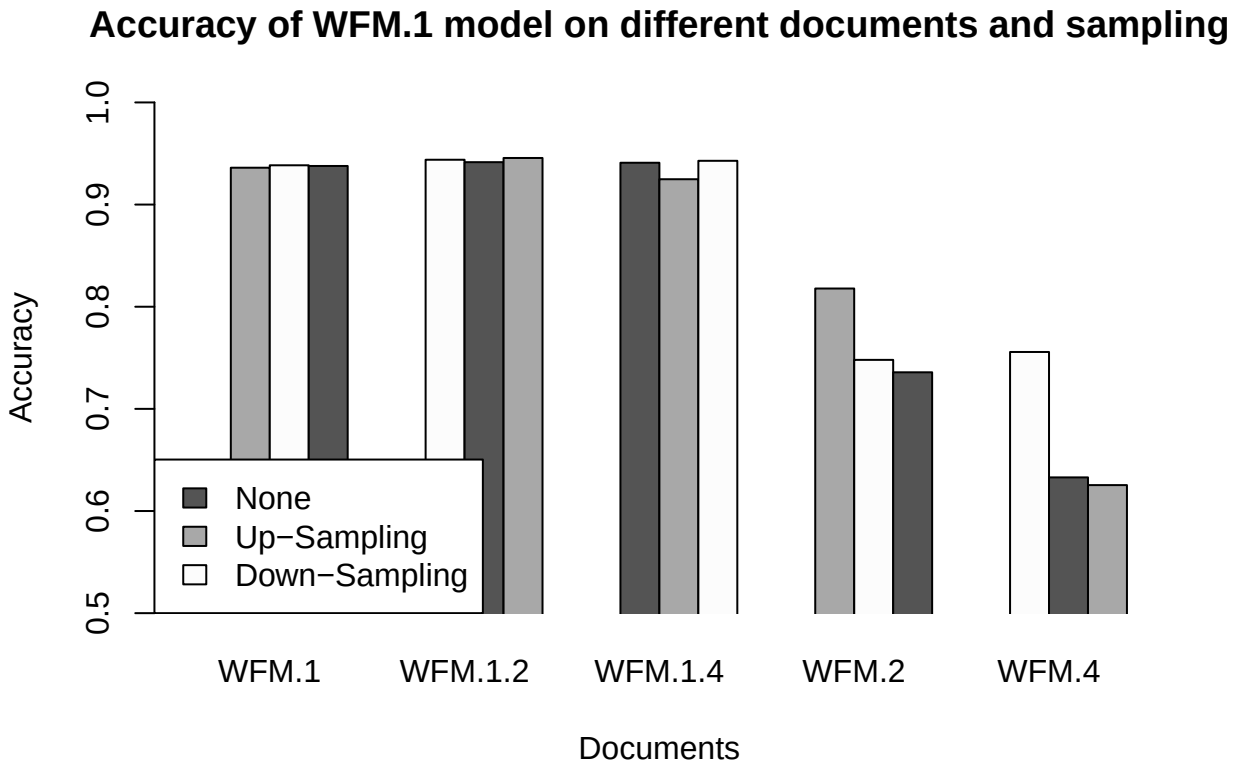
```
                    trControl = control,
                    tuneLength = length(train), ntree = 50)
    return(modelo)
}
model1 <- getModel(featureBoth1, 10)
model4 <- getModel(featureBoth4, 10)
```

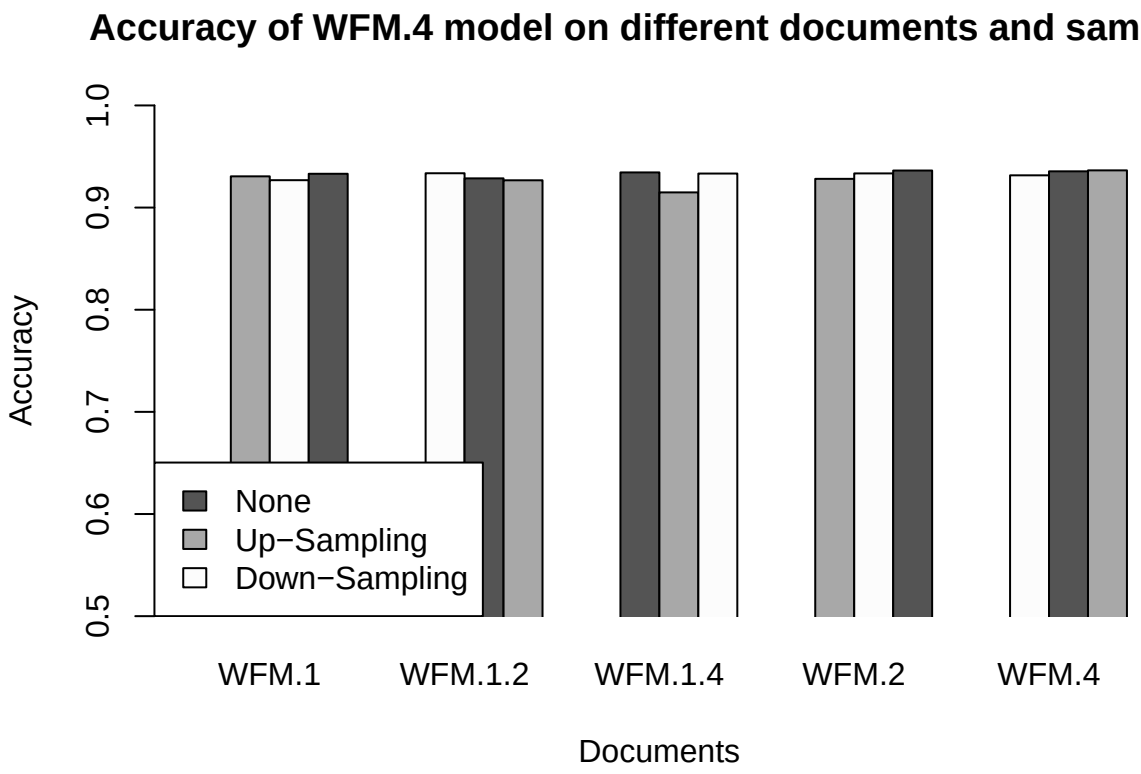
Testing of models in various documents

Similar to the previous milestone, the models will be tested on each of the other documents using Up-Sampling, Down-Sampling or any sampling technique, in order to reduce class imbalance. The function used is quite similar to the one used on the previous milestones, with the addition of a for loop that iterates between the two models created.

```
getDF <- function(mat, doc, modelo, n, lim = 10){
    data <- readMat(mat)

    unmin <- as.data.frame(data[doc])

    top10_1 <- c("a.g_P8", "Gamma_P8", "Alpha_P8", "d.g_P8", "t.g_O1",
                 "Theta_O1", "Gamma_O1", "t.g_P7", "Beta_FP1", "t.g_FP1", "Clas")
    top10_4 <- c("Alpha_P7", "Theta_O1", "Beta_P8", "Alpha_C4", "d.g_P8",
                 "Beta_C3", "d.g_O2", "Beta_P7", "a.g_C4", "Theta_C4", "Clas")

    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)

    if(mat == 'Data/Raw/FeatureMatrices.mat'){
        EDA_index <- 121:129
        unmin <- unmin[,-EDA_index]
        feature_names <- feature_names[-EDA_index]
    }else{
        feature_names <- c(feature_names, 'Clas')
    }

    colnames(unmin) <- feature_names

    if(rownames(as.data.frame(modelo$finalModel$forest$ncat))[1] == 'a.g_P8'){
        df <- unmin[,top10_1]
    }else if(rownames(as.data.frame(modelo$finalModel$forest$ncat))[1] == 'Alpha_P7'){
        df <- unmin[,top10_4]
    }

    df <- df[,c(1:n, length(df))]
    df <- filter_all(df, all_vars(. < lim))
    df <- filter_all(df, all_vars(. > -lim))
    df <- mutate(df, Clas = as.factor(Clas))
    return(df)
}
```

```r
perfDocs <- function(modelo, doc, n, samp = 'none'){
     mats <- c('Data/Raw/FeatureMatrices.mat', 'Data/Raw/FeatureMatrices_pre.mat')

     df1 <- getDF(mats[1], doc, modelo, n)
     df2 <- getDF(mats[2], doc, modelo, n)

     df <- rbind(df1, df2)

     if(samp == 'up'){
          set.seed(2031)
          df <- upSample(df[,-length(df)], df$Clas, yname = 'Clas')
     } else if(samp == 'down'){
          set.seed(2031)
          df <- downSample(df[,-length(df)], df$Clas, yname = 'Clas')
     }

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

modelos <- list(model1, model4)

for(modelo in modelos){
     n <- 10

     dfSampling <- data.frame(doc = c('WFM.1', 'WFM.1.2', 'WFM.1.4',
                                      'WFM.2', 'WFM.4'),
                              none = NA, up = NA, down = NA)

     for(doc in dfSampling$doc){
          for(boot in colnames(dfSampling)[2:length(dfSampling)]){
               dfSampling[dfSampling$doc == doc, boot] <- perfDocs(modelo, doc, n, samp = boot)
          }
     }

     if(rownames(as.data.frame(modelo$finalModel$forest$ncat))[1] == 'a.g_P8'){
             nombre <- 'WFM.1'
     }else if(rownames(as.data.frame(modelo$finalModel$forest$ncat))[1] == 'Alpha_P7'){
             nombre <- 'WFM.4'
     }
     dfSampling[] <- lapply(dfSampling, as.numeric)
     barplot(t(as.matrix.data.frame(dfSampling)), beside = TRUE,
             main = paste('Accuracy of', nombre,
                          'model on different documents and sampling'),
             col = c(gray(0.33), gray(0.66), gray(0.99)), xpd = FALSE,
             xlab = 'Documents', ylab = 'Accuracy', ylim = c(0.5, 1),
             names.arg = c('WFM.1', 'WFM.1.2', 'WFM.1.4', 'WFM.2', 'WFM.4'))
     legend('bottomleft', legend = c('None', 'Up-Sampling', 'Down-Sampling'),
            fill = c(gray(0.33), gray(0.66), gray(0.99)), bg = 'white')
}
```


Accuracy of WFM.1 model on different documents and sampling
Accuracy
0.5
0.6
0.7
0.8
0.9
1.0
None
Up−Sampling
Down−Sampling
WFM.1
WFM.1.2
WFM.1.4
WFM.2
WFM.4
Documents



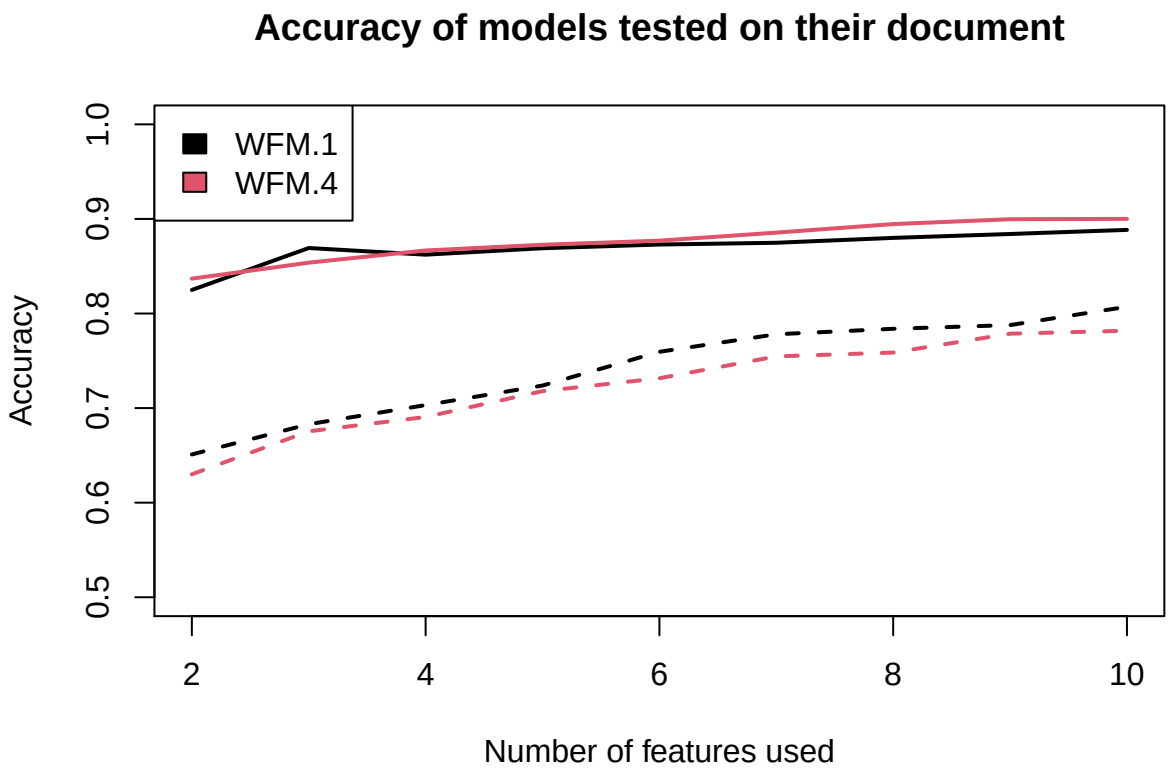
As it can be seen on the bar plots, the *WFM.4* document has more variability of data that increase the accuracy of the tree-based model on other documents. This may also indicate that the data on *WFM.4* document is included on the other documents, and so it leads to better predictions due to data leakage. Although, the current objective was to prove that the WFM.4 document is a better dataset option to train the final model on.

Model's justification on their document

The two models trained on their respective document would be tested increasing the number of features and using both rf and gbm. With the objective of validating that a parsimonious model trained on the *WFM.4* dataset could be achieved. Some of the previous milestones' code will be reused, only now the models are trained on their same document, using the previously described conditions such as the 70:30 split, the increasing cross-validations to 5-fold, and an increasing number of features from 2 to 10.

```
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{
            modelo <- train(Clas ~ ., data = training, method = met, trControl = control,
                                tuneLength = length(train))
        }

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

    n <- 5
    metodos <- c('rf', 'gbm')
    df_modelos <- data.frame(rf = NA, gbm = 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)
}

rfgbmTest <- function(df){
    n <- length(df) - 1
    dfEval <- data.frame(n_vars = 2:n, rf = NA, gbm = NA)

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

dfs <- list(featureBoth1, featureBoth4)
docs <- c('WFM.1', 'WFM.4')
plot(0,0, xlim = c(2,10), ylim = c(0.5,1), ylab = 'Accuracy',
    xlab = 'Number of features used',
    main = 'Accuracy of models tested on their document')
i = 1
for(dataset in dfs){
       df <- rfgbmTest(dataset)
       lines(df$rf ~ df$n_vars, col = i, lwd = 2, lty = 1)
       lines(df$gbm ~ df$n_vars, col = i, lwd = 2, lty = 2)
       i = i + 1
}
legend('topleft', legend = docs, fill = 1:2, bg = 'white')
```



The previous plot describes the accuracy of each model on its respective dataset, the solid line represents the rf model, while the dashed line represents the gbm model. It is worth noting that the model trained on the *WFM.4* dataset achieves a 0.90 accuracy using all of its features and a rf model, and so it can be used although there is more variability within the data.