Practical Machine Learning Project: Classification of Exercises Quality

MARIANO MOLINA GARCIA

14/06/2020

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement, a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks.

In this project, the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants will be used to quantify how well the exercise is done. The goal of your project is to predict the manner in which they did the exercise.

People were asked to perform barbell lifts correctly and incorrectly in 5 different ways, and the data previously commented have been stored in the file https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv. The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. More information about the data is available in that website, especifically in the section on the Weight Lifting Exercise Dataset.

Building the model. Data Analysis and selection of features.

The dataset contains 160 features, which can be used to predict if the exercise has been done properly or not. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl. The way the exercise has been done is labelled using de "classe" variable in the dataset. This "classe" variable can have 5 different values: A,B,C,D and E, which correspond to five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Therefore, class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

First of all, an initial analysis of the data havve been carried out. This analysis consists of:

1.- Load data. Checking the features and its range of values.

```r
library(caret)
library(AppliedPredictiveModeling)
library(dplyr)
trainingData <- read.csv("pml-training.csv")
testingData<- read.csv("pml-testing.csv")
str(trainingData)
str(testingData)
```

The number of features is 160. The number of observations for the training dataset is 19622 and the number of observations for the test dataset is 20.

```r
dim(trainingData)
```

```
## [1] 19622   160
```

```r
dim(testingData)
```

```
## [1]  20 160
```

2.- Checking if there is feature with wrong NA values. If the percentage is very high, remove the column. If it is lower, analyze the possibilities to impute values. After this process, the number of features have been reduced to 93.

```r
#Check if there are many rows of any variable with values NA
na_count <-sapply(trainingData, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
perc_na <- data.frame(100*(na_count/nrow(trainingData)))
#Remove that cols with many NA from the analysis
trainingData<-trainingData[,perc_na<90]
testingData<-testingData[,perc_na<90]
dim(trainingData)
```

```
## [1] 19622    93
```

```r
dim(testingData)
```

```
## [1] 20 93
```

3.- Checking if there is feature with empty values. If the percentage is very high, remove the column. If it is lower, analyze the possibilities to impute values.After this process, the number of features have been reduced to 60.

```r
pat <- "^[[:space:]]*$"
matches <-sapply(trainingData, function(x) grepl(pat, x))
matches<- data.frame(matches)
val_count_empty <-sapply(matches, function(y) sum(length(which(y==TRUE))))
val_count_empty<- data.frame(val_count_empty)
perc_empty <- data.frame(100*(val_count_empty/nrow(trainingData)))
#Remove that cols with many empty values from the analysis
trainingData<-trainingData[,perc_empty<90]
testingData<-testingData[,perc_empty<90]
dim(trainingData)
```

```
## [1] 19622    60
```

```r
dim(testingData)
```

```
## [1] 20 60
```

4.- Checking if there is features with a small variability of the values, which makes it useless for the prediction.There are not any feature with a single value, so the number of features for prediction is still 60.

```r
val_count_unique <-sapply(trainingData, function(y) sum(length(unique(y))))
val_count_unique<- data.frame(val_count_unique)
trainingData<-trainingData[,val_count_unique>1]
testingData<-testingData[,val_count_unique>1]
dim(trainingData)
```

```
## [1] 19622    60
```

```r
dim(testingData)
```

```
## [1] 20 60
```

5.- Checking if there are features useless for the prediction due to its meaning (time, names, etc).After removing these features, the number have been reduced to 53.

```
remove_cols = c(1:7)
trainingData<-trainingData[,-remove_cols]
testingData<-testingData[,-remove_cols]
dim(trainingData)
```

```
## [1] 19622    53
```

```
dim(testingData)
```

```
## [1] 20 53
```

6.- Remove near zero variables.

Near Zero variables: To identify these types of predictors, the following two metrics can be calculated: -the frequency of the most prevalent value over the second most frequent value (called the "frequency ratio"), which would be near one for well-behaved predictors and very large for highly-unbalanced data and -the "percent of unique values" is the number of unique values divided by the total number of samples (times 100) that approaches zero as the granularity of the data increases.

There are not any feature with near zero values, so the number of features for prediction is still 53.

```
nzv <- nearZeroVar(trainingData)
if (length(nzv) > 0) {
  trainingData <- trainingData[, -nzv]
  testingData <- testingData[, -nzv]
}
dim(trainingData)
```

```
## [1] 19622    53
```

```
dim(testingData)
```

```
## [1] 20 53
```

7.- Checking the correlation between the features to remove features with extremely high correlation, which makes the information redundant for the prediction. After analyzing the correlations, and removing the features with a correlation above 0.75, the number of features have been reduced to 32.

```
#Keep only numerical columns
numericCols <-sapply(trainingData,is.numeric)
checkCorrTrainingData <- trainingData[,numericCols]
noCheckCorrTrainingData <- trainingData[,!numericCols]
#Calculate correlation
descrCor <-  cor(checkCorrTrainingData)
summary(descrCor[upper.tri(descrCor)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.992008 -0.110080  0.002092  0.001790  0.092552  0.980924
```

```
#Remove columns with correlation > cutoff
highlyCorDescr <- findCorrelation(descrCor, cutoff = .75)
checkCorrTrainingData <- checkCorrTrainingData[,-highlyCorDescr]
descrCor2 <- cor(checkCorrTrainingData)
summary(descrCor2[upper.tri(descrCor2)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.606983 -0.103773  0.006527  0.003332  0.087527  0.736546
```

```r
#Reconstruct dataframe with no numeric columns and numeric after correlation removal
trainingData <- data.frame(checkCorrTrainingData,"classe" = noCheckCorrTrainingData)
keepProblemId <- testingData$problem_id
testingData <- testingData[,intersect(colnames(testingData),colnames(trainingData))]
testingData <- data.frame(testingData,"problem_id" = keepProblemId)
dim(trainingData)
```

```
## [1] 19622    32
```

```r
dim(testingData)
```

```
## [1] 20 32
```

8.- Check if there is outliers plotting the data with the violin shape and remove the outliers. The features gyros_dumbell, gyros_forearm_x and gyros_forearm_z show outliers, and they have been removed. After analyzing the outliers, one observation of the training data have been removed, having 19621 to perform the analysis.

```r
library(dplyr)
trainingData<-trainingData%>%filter(gyros_dumbbell_y<10)
trainingData<-trainingData%>%filter(gyros_forearm_x>-10)
trainingData<-trainingData%>%filter(gyros_forearm_z<10)
dim(trainingData)
```

```
## [1] 19621    32
```

```r
dim(testingData)
```

```
## [1] 20 32
```

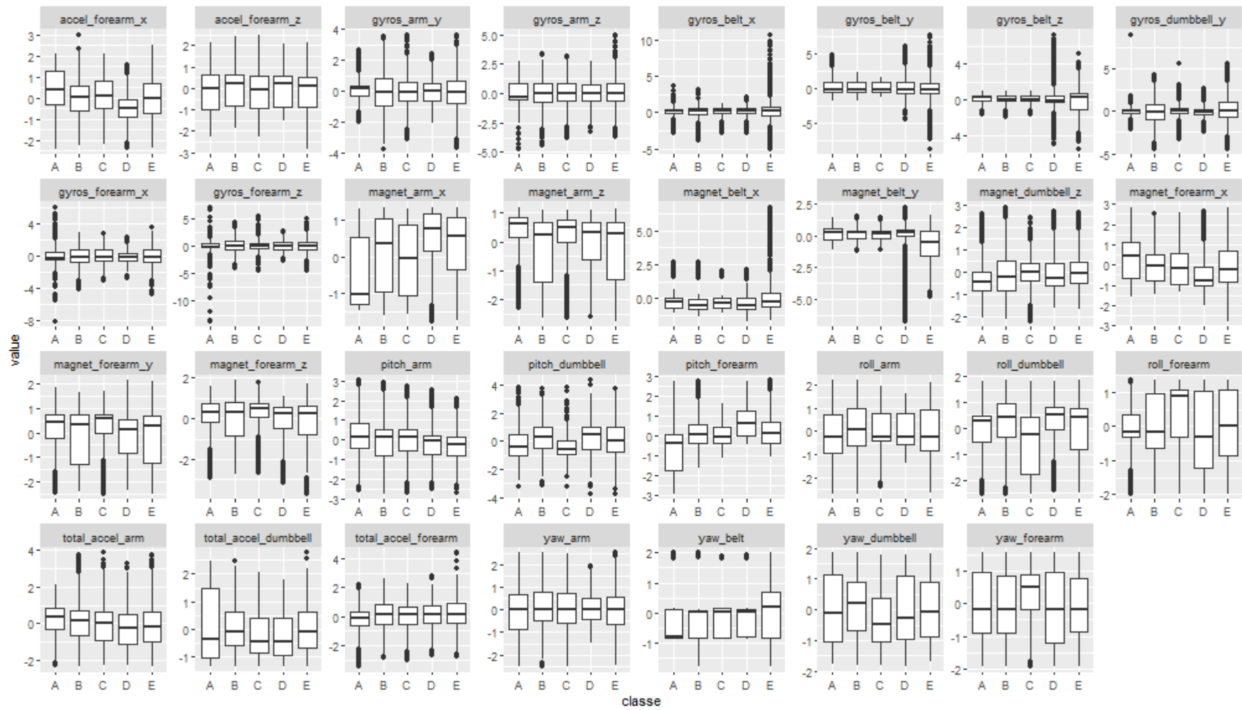
9.- Applying a preprocess to the data, to improve the performance or the prediction algorithms. From all the possible preprocess options (Possible values are “BoxCox”, “YeoJohnson”, “expoTrans”, “center”, “scale”, “range”, “knnImpute”, “bagImpute”, “medianImpute”, “pca”, “ica”, “spatialSign”, “corr”, “zv”, “nzv”, and “conditionalX”) centering and scaling have been selected. The correlation and nzv analysis have been carried out independently in previous steps.

```r
#Centering and Scaling
preProcValues <- preProcess(trainingData, method = c("center", "scale"))
trainTransformed <- predict(preProcValues, trainingData)
testTransformed <- predict(preProcValues, testingData)
```

10.- Finally, some plots are performed to analyze the reamining features before proceeding to the use of multiclass classification models for prediction.

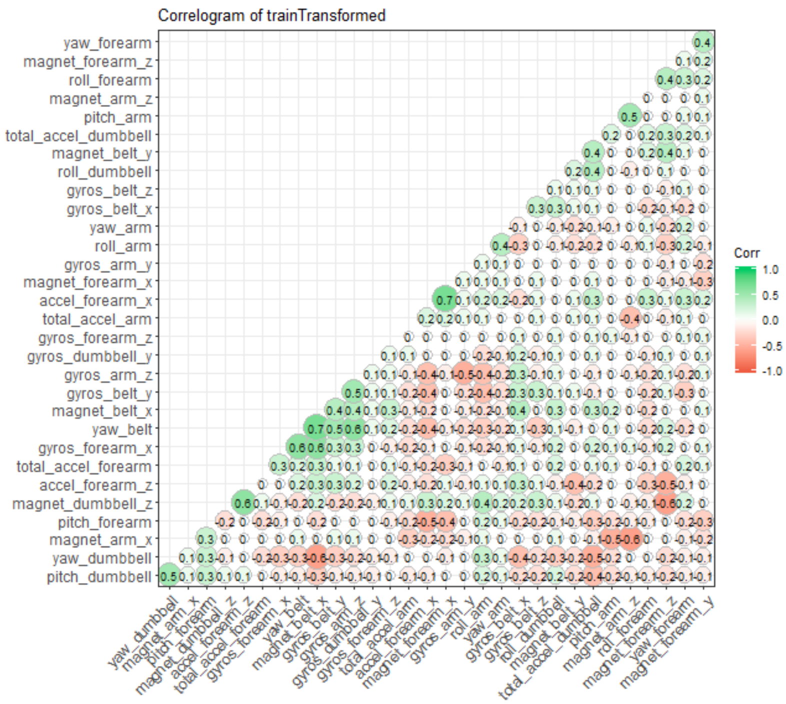
A boxplot analysis have been carried to, to visualize if there are some features which can be detected at first sight as good feature to distinguised between “classe” groups.

```r
library(ggplot2)
library(tidyr)
trainTransformedlong <- gather(trainTransformed, key="measure", value="value"
                               ,c(names(trainTransformed[,1:31])))
p <- ggplot(data = trainTransformedlong, aes(x=classe,y=value)) +
    geom_boxplot()
p + facet_wrap( ~measure, scales="free",ncol=8)
```



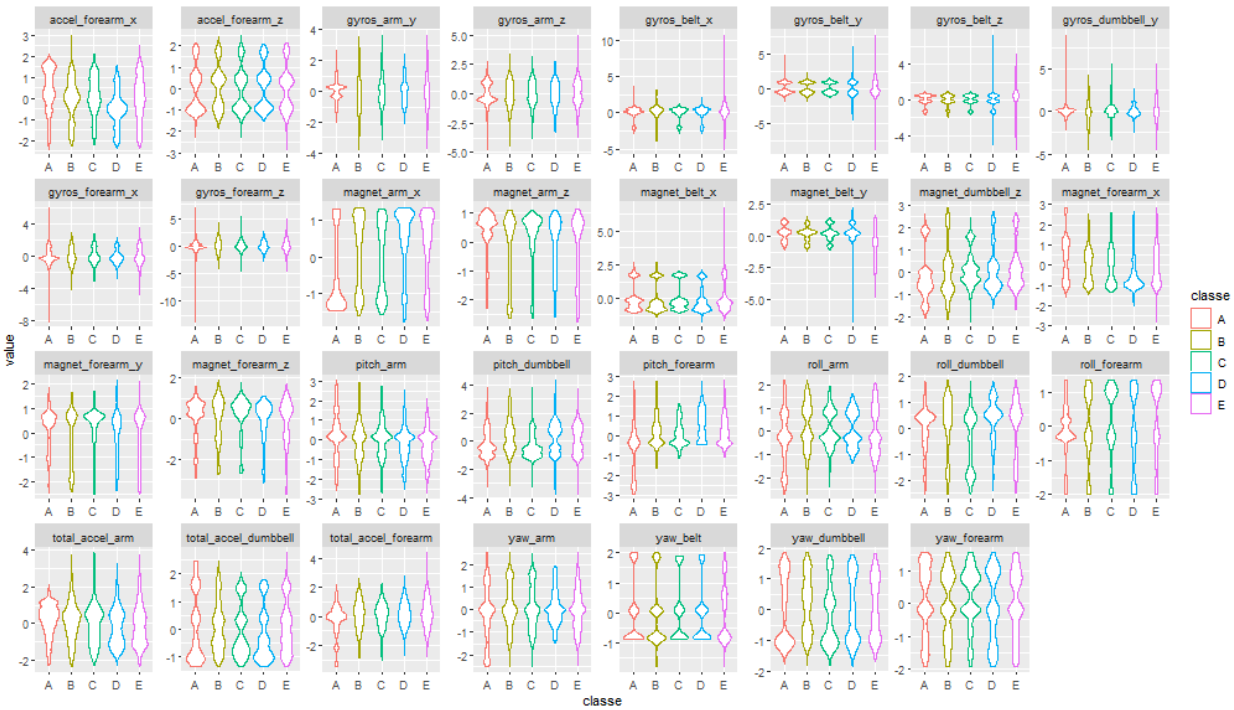
After that, a correlogram to show the correlation between the remaining features is presented as well.

```r
#CorreloGram
library(ggplot2)
library(ggcorrplot)
corr <- round(cor(trainTransformed[,1:31]), 1)
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of trainTransformed",
           ggtheme=theme_bw)
```



Finally, a violin plot to see the distribution of values per "classe" for each feature is shown as well.

```
library(ggplot2)
library(tidyr)
library(dplyr)
trainTransformedlong3 <- gather(trainTransformed, key="measure", value="value",
                                c(names(trainTransformed[,1:31])))
val=colnames(trainTransformedlong3)[1]
# plot
g <- ggplot(trainTransformedlong3, aes(x=classe, y=value,color=classe))
g <- g + geom_violin()
g + facet_wrap( ~measure,scales="free",ncol=8)
```



Building the model. Cross Validation and Analysis of Multiclass classification algorithms.

In this section, the cross validation process and the analysis of the multilass classification algorithms are performed.

To perform cross validation, the training data into a pure training dataset (80%) and a validation dataset (20%). The validation set will be used to assess the accuracy of the prediction of the classifier and the out-of-sample error.

The following 15 multiclass classification methos have been tested, and its accuracy compared: knn, rf, pda, parRF, sda, hdrda, LMT, slda, hdda, RRFglobal, C5.0, LogitBoost, pam, PART, rpart.

For all of them, a k-fold cross-validation method has been selected, with k = 10, which is the most recommended k value. The training sample will be randomly partitioned into 10 equal sized subsamples. A single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data.The cross-validation process is repeated 10 times, with each of the 10 subsamples used exactly once as the validation data.

Other methods for cross validation could have been selected. The available resampling methods are: The "boot", "boot632", "optimism_boot", "boot_all", "cv", "repeatedcv", "LOOCV", "LGOCV" (for repeated training/test splits), "none" (only fits one model to the entire training set), "oob" (only for random forest, bagged trees, bagged earth, bagged flexible discriminant analysis, or conditional tree forest models), timeslice, "adaptive_cv", "adaptive_boot" or "adaptive_LGOCV"

First of all, we create the training and validation sets. 15699 observations are used for training and 3145 are used for validation.

```
set.seed(12345)
partitionSamples <- createDataPartition(y=trainTransformed$classe, p=0.8, list=FALSE)
trainTransformed <- trainTransformed[partitionSamples, ]
validationTransformed <- trainTransformed[-partitionSamples, ]
```

```r
dim(trainTransformed)
```

```
## [1] 15699    32
```

```r
dim(validationTransformed)
```

```
## [1] 3145   32
```

```r
library(doParallel)
registerDoParallel(4)
getDoParWorkers()
```

- KNN

```r
fitKNN = train(
    classe ~.,
    data = trainTransformed,
    method = "knn",
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE),
    tuneGrid = expand.grid(k = seq(1, 21, by = 2))
)
head(predict(fitKNN, type = "prob"))
predictedKNN<-predict(fitKNN,newdata=trainTransformed)
CM_fitKNN <- confusionMatrix(predictedKNN, trainTransformed$classe)
predictedKNN2<-predict(fitKNN,validationTransformed)
CM_fitKNN_validation <- confusionMatrix(predictedKNN2, validationTransformed$classe)
varImp_fitKNN <- varImp(fitKNN)
```

- Random Forest

```r
fitRF = train(
    classe ~.,
    data = trainTransformed,
    method = 'rf',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitRF, type = "prob"))
predictedRF<-predict(fitRF,newdata=trainTransformed)
CM_fitRF <- confusionMatrix(predictedRF, trainTransformed$classe)
predictedRF2<-predict(fitRF,validationTransformed)
CM_fitRF_validation <- confusionMatrix(predictedRF2, validationTransformed$classe)
varImp_fitRF <- varImp(fitRF)
```

- pda

```r
library(mda)
fitpda= train(
    classe ~.,
    data = trainTransformed,
    method = 'pda',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitpda, type = "prob"))
predictedpda<-predict(fitpda,newdata=trainTransformed)
```

```
CM_fitpda <- confusionMatrix(predictedpda, trainTransformed$classe)
predictedpda2<-predict(fitpda,validationTransformed)
CM_fitpda_validation <- confusionMatrix(predictedpda2, validationTransformed$classe)
varImp_fitpda <- varImp(fitpda)
```

- Parallel Random Forest

```
library(e1071)
library(randomForest)
fitPRF = train(
    classe ~.,
    data = trainTransformed,
    method = 'parRF',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitPRF, type = "prob"))
predictedPRF<-predict(fitPRF,newdata=trainTransformed)
CM_fitPRF <- confusionMatrix(predictedPRF, trainTransformed$classe)
predictedPRF2<-predict(fitPRF,validationTransformed)
CM_fitPRF_validation <- confusionMatrix(predictedPRF2, validationTransformed$classe)
varImp_fitPRF <- varImp(fitPRF)
```

- sda

```
library(sda)
fitsda= train(
    classe ~.,
    data = trainTransformed,
    method = 'sda',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitsda, type = "prob"))
predictedsda<-predict(fitsda,newdata=trainTransformed)
CM_fitsda <- confusionMatrix(predictedsda, trainTransformed$classe)
predictedsda2<-predict(fitsda,validationTransformed)
CM_fitsda_validation <- confusionMatrix(predictedsda2, validationTransformed$classe)
varImp_fitsda <- varImp(fitsda)
```

- hdrda

```
library(sparsediscrim)
fithdrda= train(
    classe ~.,
    data = trainTransformed,
    method = 'hdrda',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fithdrda, type = "prob"))
predictedhdrda<-predict(fithdrda,newdata=trainTransformed)
CM_fithdrda <- confusionMatrix(predictedhdrda, trainTransformed$classe)
predictedhdrda2<-predict(fithdrda,validationTransformed)
CM_fithdrda_validation <- confusionMatrix(predictedhdrda2, validationTransformed$classe)
```

```
varImp_fithdrda <- varImp(fithdrda)
```

- LMT (Logistic Model Trees)

```
##Logistic Model Trees
library(RWeka)
fitLMT = train(
    classe ~.,
    data = trainTransformed,
    method = "LMT",
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitLMT, type = "prob"))
predictedLMT<-predict(fitLMT,newdata=trainTransformed)
CM_fitLMT <- confusionMatrix(predictedLMT, trainTransformed$classe)
predictedLMT2<-predict(fitLMT,validationTransformed)
CM_fitLMT_validation <- confusionMatrix(predictedLMT2, validationTransformed$classe)
varImp_fitLMT <- varImp(fitLMT)
```

- slda

```
library(ipred)
fitslda= train(
    classe ~.,
    data = trainTransformed,
    method = 'slda',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitslda, type = "prob"))
predictedslda<-predict(fitslda,newdata=trainTransformed)
CM_fitslda <- confusionMatrix(predictedslda, trainTransformed$classe)
predictedslda2<-predict(fitslda,validationTransformed)
CM_fitslda_validation <- confusionMatrix(predictedslda2, validationTransformed$classe)
varImp_fitslda <- varImp(fitslda)
```

- hdda

```
library(HDclassif)
fithdda= train(
    classe ~.,
    data = trainTransformed,
    method = 'hdda',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fithdda, type = "prob"))
predictedhdda<-predict(fithdda,newdata=trainTransformed)
CM_fithdda <- confusionMatrix(predictedhdda, trainTransformed$classe)
predictedhdda2<-predict(fithdda,validationTransformed)
CM_fithdda_validation <- confusionMatrix(predictedhdda2, validationTransformed$classe)
varImp_fithdda <- varImp(fithdda)
```

- RRFglobal

```r
library(RRF)
fitRRFglobal= train(
    classe ~.,
    data = trainTransformed,
    method = 'RRFglobal',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                              ,allowParallel = TRUE)
)
head(predict(fitRRFglobal, type = "prob"))
predictedRRFglobal<-predict(fitRRFglobal,newdata=trainTransformed)
CM_fitRRFglobal <- confusionMatrix(predictedRRFglobal, trainTransformed$classe)
predictedRRFglobal2<-predict(fitRRFglobal,validationTransformed)
CM_fitRRFglobal_validation <- confusionMatrix(predictedRRFglobal2, validationTransformed$classe)
varImp_fitRRFglobal <- varImp(fitRRFglobal)
```

- C5.0

```r
fitC50 = train(
    classe ~.,
    data = trainTransformed,
    method = 'C5.0',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                              ,allowParallel = TRUE)
)
head(predict(fitC50, type = "prob"))
predictedC50<-predict(fitC50,newdata=trainTransformed)
CM_fitC50 <- confusionMatrix(predictedC50, trainTransformed$classe)
predictedC502<-predict(fitC50,validationTransformed)
CM_fitC50_validation <- confusionMatrix(predictedC502, validationTransformed$classe)
varImp_fitC50 <- varImp(fitC50)
```

- LogitBoost

```r
library(caTools)
fitLogitBoost= train(
    classe ~.,
    data = trainTransformed,
    method = 'LogitBoost',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                              ,allowParallel = TRUE)
)
head(predict(fitLogitBoost, type = "prob"))
predictedLogitBoost<-predict(fitLogitBoost,newdata=trainTransformed)
CM_fitLogitBoost <- confusionMatrix(predictedLogitBoost, trainTransformed$classe)
predictedLogitBoost2<-predict(fitLogitBoost,validationTransformed)
CM_fitLogitBoost_validation <- confusionMatrix(predictedLogitBoost2, validationTransformed$classe)
varImp_fitLogitBoost <- varImp(fitLogitBoost)
```

- pam

```r
library(pamr)
fitpam= train(
    classe ~.,
    data = trainTransformed,
    method = 'pam',
```

```
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitpam, type = "prob"))
predictedpam<-predict(fitpam,newdata=trainTransformed)
CM_fitpam <- confusionMatrix(predictedpam, trainTransformed$classe)
predictedpam2<-predict(fitpam,validationTransformed)
CM_fitpam_validation <- confusionMatrix(predictedpam2, validationTransformed$classe)
varImp_fitpam <- varImp(fitpam)
```

- PART (Rule Based Classifier)

```
fitRBC = train(
    classe ~.,
    data = trainTransformed,
    method = 'PART',
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitRBC, type = "prob"))
predictedRBC<-predict(fitRBC,newdata=trainTransformed)
CM_fitRBC <- confusionMatrix(predictedRBC, trainTransformed$classe)
predictedRBC2<-predict(fitRBC,validationTransformed)
CM_fitRBC_validation <- confusionMatrix(predictedRBC2, validationTransformed$classe)
varImp_fitRBC <- varImp(fitRBC)
```

- rpart (Tree with RPART)

```
fitRPART = train(
    classe ~.,
    data = trainTransformed,
    method = "rpart",
    trControl = trainControl(method = "cv", number = 10,savePredictions = "final"
                             ,allowParallel = TRUE)
)
head(predict(fitRPART, type = "prob"))
predictedRPART<-predict(fitRPART,newdata=trainTransformed)
CM_fitRPART <- confusionMatrix(predictedRPART, trainTransformed$classe)
predictedRPART2<-predict(fitRPART,validationTransformed)
CM_fitRPART_validation <- confusionMatrix(predictedRPART2, validationTransformed$classe)
varImp_fitRPART <- varImp(fitRPART)
```

Results of the multiclass classification models. Importance of features.

In the following table, the results for accuracy for the algorithms evaluated are presented:

```
result_Analysis = data.frame(methods = methods<-c("C50","hdda","hdrda","KNN","LMT"
                                                  ,"LogitBoost","pam","pda","PRF"
                                                  ,"RBC","RF","RPART","RRFglobal"
                                                  ,"sda","slda"),
InSampleAccuracy = c(CM_fitC50$overall["Accuracy"],CM_fithdda$overall["Accuracy"]
             ,CM_fithdrda$overall["Accuracy"],CM_fitC50$overall["Accuracy"],
             CM_fitKNN$overall["Accuracy"],CM_fitLogitBoost$overall["Accuracy"]
```

```r
             ,CM_fitpam$overall["Accuracy"],CM_fitpda$overall["Accuracy"],
             CM_fitPRF$overall["Accuracy"],CM_fitRBC$overall["Accuracy"]
             ,CM_fitRF$overall["Accuracy"],CM_fitRPART$overall["Accuracy"],
             CM_fitRRFglobal$overall["Accuracy"],CM_fitsda$overall["Accuracy"]
             ,CM_fitslda$overall["Accuracy"]),
OutSampleAccuracy = c(CM_fitC50_validation$overall["Accuracy"]
                      ,CM_fithdda_validation$overall["Accuracy"]
             ,CM_fithdrda_validation$overall["Accuracy"]
             ,CM_fitC50_validation$overall["Accuracy"],
             CM_fitKNN_validation$overall["Accuracy"]
             ,CM_fitLogitBoost_validation$overall["Accuracy"]
             ,CM_fitpam_validation$overall["Accuracy"]
             ,CM_fitpda_validation$overall["Accuracy"],
             CM_fitPRF_validation$overall["Accuracy"]
             ,CM_fitRBC_validation$overall["Accuracy"]
             ,CM_fitRF_validation$overall["Accuracy"]
             ,CM_fitRPART_validation$overall["Accuracy"],
             CM_fitRRFglobal_validation$overall["Accuracy"]
             ,CM_fitsda_validation$overall["Accuracy"]
             ,CM_fitslda_validation$overall["Accuracy"]))
result_Analysis
```

```
##        methods InSampleAccuracy OutSampleAccuracy
## 1          C50        1.0000000         1.0000000
## 2         hdda        0.7606217         0.7472178
## 3        hdrda        0.8246385         0.8146264
## 4          KNN        1.0000000         1.0000000
## 5          LMT        1.0000000         1.0000000
## 6   LogitBoost        0.8753964         0.8702202
## 7          pam        0.4237850         0.4181240
## 8          pda        0.5866616         0.5860095
## 9          PRF        1.0000000         1.0000000
## 10         RBC        0.9958596         0.9936407
## 11          RF        1.0000000         1.0000000
## 12       RPART        0.5296516         0.5240064
## 13   RRFglobal        1.0000000         1.0000000
## 14         sda        0.5864068         0.5860095
## 15        slda        0.4249315         0.4095390
```

The results show that 6 methods have both an in sample accuracy and an out of sample accuracy of 1, meaning that they are able to classify properly all the observations in the training data and in the validation data. With this accuracies, it is expected to have a very high classification out of sample accuracy when analyzing new data coming from the devices.

To finalize the analysis,the importance of each predictor have been analyzed for those 6 methods with a 100% accuracy.

- Importance for C50

```
## C5.0 variable importance
##
##   only 20 most important variables shown (out of 31)
##
##                    Overall
## magnet_belt_y       100.00
## magnet_forearm_x    100.00
```

```
## magnet_dumbbell_z  100.00
## gyros_belt_z       100.00
## yaw_belt           100.00
## magnet_forearm_y   100.00
## pitch_forearm      100.00
## roll_arm            99.79
## gyros_belt_y        99.62
## yaw_arm             99.21
## magnet_belt_x       98.97
## gyros_dumbbell_y    98.70
## gyros_arm_y         97.10
## roll_dumbbell       97.03
## pitch_arm           95.22
## magnet_forearm_z    93.79
## accel_forearm_z     92.19
## magnet_arm_z        87.17
## gyros_forearm_x     85.57
## gyros_belt_x        84.61
```

- Importance for KNN

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 31)
##
##                          A       B      C      D      E
## pitch_forearm       62.981 100.000 67.289 62.981 100.00
## accel_forearm_x     42.054  81.878 42.054 42.054  81.88
## magnet_arm_x        53.331  78.445 64.563 53.331  78.44
## magnet_forearm_x    39.685  71.112 34.651 34.651  71.11
## pitch_dumbbell      51.215  51.215 51.215 69.938  43.45
## magnet_belt_y       12.563   8.953 67.065  8.953  12.56
## roll_dumbbell       38.703  51.296 30.393 60.847  51.30
## magnet_dumbbell_z   54.698  35.417 52.636 22.749  54.70
## magnet_arm_z        51.538  51.538 51.538 51.538  39.38
## pitch_arm           25.817  42.131 47.726 25.817  42.13
## total_accel_arm     30.507  41.637 32.015 13.950  41.64
## yaw_dumbbell        18.917  18.917 18.917 39.479  11.93
## magnet_forearm_y    18.665  36.967 28.164 23.190  36.97
## roll_forearm        35.467   5.398 12.624 25.151  35.47
## roll_arm            34.348  34.348 34.348 34.348  23.13
## total_accel_forearm 24.505  27.498 32.567 24.505  27.50
## magnet_belt_x       24.386  24.386 24.386 24.386  18.51
## yaw_forearm         10.355  16.556 11.327 22.347  16.56
## magnet_forearm_z    20.176  15.469 11.693 15.002  20.18
## gyros_belt_z         9.781  17.952  9.781  9.781  17.95
```

- Importance for LMT

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 31)
##
##                          A       B      C      D      E
```

```
## pitch_forearm       62.981 100.000 67.289 62.981 100.00
## accel_forearm_x     42.054  81.878 42.054 42.054  81.88
## magnet_arm_x        53.331  78.445 64.563 53.331  78.44
## magnet_forearm_x    39.685  71.112 34.651 34.651  71.11
## pitch_dumbbell      51.215  51.215 51.215 69.938  43.45
## magnet_belt_y       12.563   8.953 67.065  8.953  12.56
## roll_dumbbell       38.703  51.296 30.393 60.847  51.30
## magnet_dumbbell_z   54.698  35.417 52.636 22.749  54.70
## magnet_arm_z        51.538  51.538 51.538 51.538  39.38
## pitch_arm           25.817  42.131 47.726 25.817  42.13
## total_accel_arm     30.507  41.637 32.015 13.950  41.64
## yaw_dumbbell        18.917  18.917 18.917 39.479  11.93
## magnet_forearm_y    18.665  36.967 28.164 23.190  36.97
## roll_forearm        35.467   5.398 12.624 25.151  35.47
## roll_arm            34.348  34.348 34.348 34.348  23.13
## total_accel_forearm 24.505  27.498 32.567 24.505  27.50
## magnet_belt_x       24.386  24.386 24.386 24.386  18.51
## yaw_forearm         10.355  16.556 11.327 22.347  16.56
## magnet_forearm_z    20.176  15.469 11.693 15.002  20.18
## gyros_belt_z         9.781  17.952  9.781  9.781  17.95
```

- Importance for PRF

```
## parRF variable importance
##
##   only 20 most important variables shown (out of 31)
##
##                      Overall
## yaw_belt              100.00
## magnet_dumbbell_z      76.92
## magnet_belt_y          70.57
## pitch_forearm          70.13
## roll_dumbbell          53.55
## roll_forearm           52.66
## gyros_belt_z           44.60
## roll_arm               41.83
## total_accel_dumbbell   41.31
## yaw_dumbbell           38.30
## gyros_dumbbell_y       36.29
## magnet_arm_x           33.88
## pitch_dumbbell         32.60
## magnet_forearm_z       32.03
## accel_forearm_x        30.22
## magnet_belt_x          29.82
## yaw_arm                29.11
## magnet_forearm_y       28.88
## accel_forearm_z        26.76
## magnet_forearm_x       26.13
```

- Importance for RF

```
## rf variable importance
##
##   only 20 most important variables shown (out of 31)
##
##                      Overall
```

```
## yaw_belt              100.00
## magnet_dumbbell_z      71.50
## magnet_belt_y          65.76
## pitch_forearm          64.84
## roll_forearm           55.40
## roll_dumbbell          53.84
## roll_arm               40.63
## gyros_belt_z           38.86
## total_accel_dumbbell   36.65
## yaw_dumbbell           35.84
## gyros_dumbbell_y       34.23
## accel_forearm_x        32.07
## magnet_arm_x           32.01
## magnet_forearm_z       31.40
## pitch_dumbbell         29.41
## magnet_belt_x          28.04
## accel_forearm_z        27.19
## yaw_arm                26.86
## magnet_forearm_y       26.62
## magnet_forearm_x       24.06
```

- Importance for RRF Global

```
## RRFglobal variable importance
##
##   only 20 most important variables shown (out of 31)
##
##                       Overall
## yaw_belt              100.00
## pitch_forearm          85.24
## magnet_belt_y          78.94
## magnet_dumbbell_z      58.06
## roll_dumbbell          52.50
## total_accel_dumbbell   40.50
## roll_forearm           36.92
## gyros_belt_z           31.47
## accel_forearm_z        31.36
## magnet_forearm_z       20.90
## accel_forearm_x        19.69
## magnet_belt_x          18.83
## yaw_dumbbell           16.53
## roll_arm               16.30
## yaw_arm                15.89
## magnet_arm_x           12.65
## pitch_dumbbell         12.25
## gyros_arm_y            10.20
## pitch_arm              10.05
## gyros_dumbbell_y        9.31
```