Machine learning algorithm to predict how well barbel lifts are being made

EXECUTIVE SUMMARY

This document describes a model for the prediction of the way in which a barber lift is performed based in different measures got from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in five different ways in order to realize this study. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har.

For the purpose of this assignment a training and a testing sets were providing.

Once created the model from 20% of the training set, the result obtained can be seen in the Appendix (Figure 1). Here it can be seen the expected out of sample error.

Also, in Appendix (Figure 2), it can be seen the prediction for the 20 different test cases.

MODEL CREATION

DATA HANDLING

Two different sets will be used, one for training and one for the 20 different test cases

```
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
```

VARIABLES ANALYSIS

We will first drop all variables that in the test have all NAs or blanks, and then those that are not related to the accelerators measures, like name of the user, time, ... (the first 7 columns)

```
test <- test[,colSums(is.na(test))<nrow(test)]
valid_cols <- c(names(test)[8:length(names(test))], "classe")
train <- train[, names(train) %in% valid_cols]
```

PARTITION

For the creation of the model, the training test will be divided in two subsets: - 60% for the training - 40% for its testing

```
## Partition
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
train1_index <- createDataPartition(train$classe, p=0.6, list=FALSE)
train1 <- train[train1_index,]
test1 <- train[-train1_index,]
```

TRAINING

The model will used the Random Forest method.

As there are different type of errors in the performance of the exercise, we don't expect a linear model.

```
## Random Forest
modFit <- train(train1$classe~., data=train1, method="rf")
```

```
## Loading required package: randomForest
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
modFit
```

```
## Random Forest
##
## 11776 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     1         1      0.002        0.003
##   30    1         1      0.002        0.003
##   50    1         1      0.005        0.006
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

TESTING

Once, we have created the model, we test it with the remainder 40% of the training subset

```
predictions <- predict(modFit, newdata = test1)
```

APPENDIX

FIGURE 1

```
confusionMatrix(predictions, test1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   10    0    0    0
##          B    1 1505   10    0    1
##          C    3    3 1349   26    1
##          D    0    0    9 1259    6
##          E    1    0    0    1 1434
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.988, 0.993)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.988
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.998    0.991    0.986    0.979    0.994
## Specificity             0.998    0.998    0.995    0.998    1.000
## Pos Pred Value          0.996    0.992    0.976    0.988    0.999
## Neg Pred Value          0.999    0.998    0.997    0.996    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.172    0.160    0.183
## Detection Prevalence    0.285    0.193    0.176    0.162    0.183
## Balanced Accuracy       0.998    0.995    0.991    0.988    0.997
```

FIGURE 2

```
predict(modFit, newdata=test)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```