Prediction Models: Quality of Exercising

Paloma Urbano

October 18, 2016

Summary

This project predicts the manner in which a set of individuals exercise. We use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to create a predictive model. The data used is part of the http://groupware.les.inf.puc-rio.br/har effort.

The model was built using random forest and cross validated using the confusionMatrix command in R. The estimated out of sample error is 0.69%.

The model shows 0.991164 of accuracy and it was able to predict the 20 different test cases of the test data set.

Data Preparation

After loading the 2 data sets, training and testing, we proceed to prepare the data for the model fitting.

We took the 3 following steps:

- 1.- remove first seven attributes as they were not relevant for the model fitting
- 2.- remove missing data(NAs)
- 3.- the near zero values

```
# Loading training and testing data sets
training <- read.csv("~/R Code/pml-training.csv")
training$classe <- as.factor(training$classe)
testing <- read.csv("~/R Code/pml-testing.csv")
# Set the seed for repeatibility
set.seed(250)

#Remove unnecessary columns
training <- training[ , -(1:7)]
testing <- testing[ , -(1:7)]

# Remove missing values
t_Naind <- apply(training,2, function(x){sum(is.na(x))})
trainCL <- training[,which(t_Naind==0)]

te_Naind <- apply(testing,2,function(x){sum(is.na(x))})
testCL <- testing[,which(te_Naind==0)]
```

```
# Remove the new zero values
t_nz <- nearZeroVar(trainCL, saveMetrics=TRUE)
trainCLn <- trainCL[,which(t_nz$nzv==FALSE)]

te_nz <- nearZeroVar(testCL, saveMetrics=TRUE)
testCLn <- testCL[,which(te_nz$nzv==FALSE)]
```

The prepared data sets results on the following dimensions:

- Training data set: 19622, 53
- Test data set: 20, 53

Model Creation

We took the training data set and divided into 2 sets:

- 1.- 70% of the training data becomes our training data set
- 2.- 30% of the training data will be used to test the model

We use random forest to fit our model.

```
# Divide the training data

inTrain <- createDataPartition(y=trainCLn$classe, p=0.7, list=FALSE)
trainA <- trainCLn[inTrain,]
train_test <- trainCLn[-inTrain,]


#modFit <- train(classe ~., method="rf", data = trainA, importance=TRUE,
allowparallel=TRUE)

load("~/model.rda")
```

Model Validation

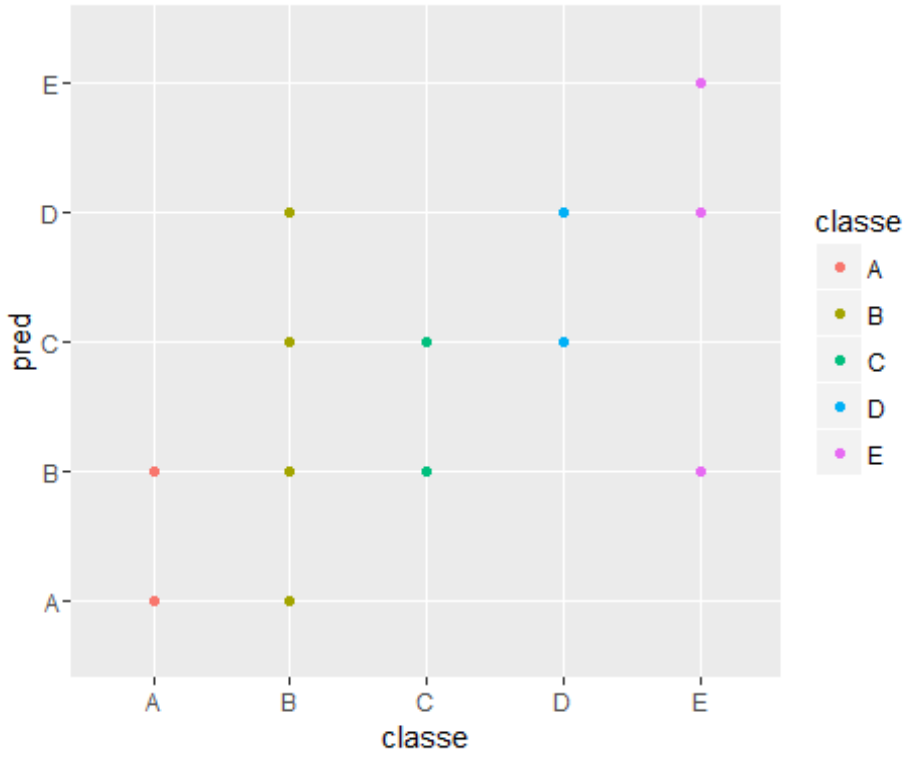
The model was validated showing the following accuracy measurements:

```
# Validate model
confusionMatrix(train_test$classe, predict(modFit, train_test))$overall

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.2.5

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9972812      0.9965611      0.9955886      0.9984452      0.2846219
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

This plot is a visual validation of the fitted model

```
# Visual model fitting validation

pred <- predict(modFit, train_test)
qplot(classe, pred, colour=classe, data=train_test)
```



Run Prediction

Considering the high level of accuracy we used this model to predict the test data set.Results of the prediction are show here:

```r
# Predict the test data set
pred1 <- predict(modFit, testCLn)

pred1
```

```
##  [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
```

Summary

This exercise goes through all steps of fitting a predictive model. Creating models with the data set proved very computational intensive. I was surprised of the high accuracy resulted out of just one model.