Exercise Manner Prediction

Hugo Barros

November 10, 2017

R Markdown

The goal of this assignment is to predict the manner in which an exercise was performed.

For that, we need to make sure that the training and test sets have some common rules, i.e., perform some data cleaning.

We have a training and a test dataset. So our first task will be to read the two data sets(treating empty cells as NAs), and join them, without the last column of each (because they are different and also because they're not meant to go through data cleaning) into one dataframe.

```
suppressPackageStartupMessages(library(randomForest))
suppressPackageStartupMessages(library(doParallel))
suppressPackageStartupMessages(library(caret))

traindat <- read.csv("c:/Users/Dafh/Documents/R/Work/Machine Learning/trainDat.csv", na.strings = c("",

testdat <- read.csv("c:/Users/Dafh/Documents/R/Work/Machine Learning/testDat.csv", na.strings = c("",NA

joint <- rbind(traindat[,-160], testdat[,-160])
```

Now we will remove columns with only zero-values and those where 90% or more of their values are NAs.

```
pre_joint <- preProcess(joint,method = c("zv","corr"))
joint1 <- predict(pre_joint,joint)
anyNA(joint1)
```

```
## [1] TRUE
```

```
na_count <- sapply(joint1, function(y) sum(is.na(y)))

zeroNA <- joint1[,(na_count/nrow(joint1)) < 0.90]
```

Now we have a dataframe w/out columns composed of only zero-values, or mostly of NAs, which makes us ready to split the dataframe back into a training and a test datasets. We will also add the last column of training data(classe) that we removed when we joined the two datasets

```
trClean <- zeroNA[1:19622,]
testClean <- zeroNA[19623:19642,]
trClean$classe <- traindat$classe
```

On the train dataframe, it is imperative to do some variable selection in order to get rid of some of the noise of variables that are not likely to be related to the outcome. Some of these variables are: X, user_name, the timestamps(all three of them).

```
trClean <- trClean[,-c(1:6)]
```

We consider our training set clean enough for a fair shot at modeling and accurate predictions.

Four types of model will be created and applied on the train set and then on the test set: lda, rpart, a combination of the previous two, and radomForrest.

LDA

(lda modeling, prediction on train data and then test data)

```
 tail(colnames(trClean))
```

```
## [1] "accel_forearm_y"  "accel_forearm_z"  "magnet_forearm_x"
## [4] "magnet_forearm_y" "magnet_forearm_z" "classe"
```

```
#head(trClean[,"classe"])
ldamod <- train(classe ~ ., data = trClean, method="lda")
confusionMatrix(predict(ldamod,trClean),trClean[,46])$overall[1]
```

```
##  Accuracy
## 0.6840281
```

RPART

(rpart modeling, prediction on train data and then test data)

```
rpartmod <- train(classe ~ ., data = trClean, method="rpart")
confusionMatrix(predict(rpartmod,trClean),trClean[,"classe"])$overall[1]
```

```
##  Accuracy
## 0.5292529
```

As far as accuracy on the train data, lda is performing better than rpart. Still, both models are not very accurate even on the data where they were trained. However, that does not mean it won't perform better on the test data.

RandomForrest modeling

we will apply a couple of tweaks to randomForrests in order to make it less computationally demanding.(EXPLAIN TWEAKS LIVE CV ON RANDOMFORREST)

```
my_cluster <- makeCluster(detectCores()-1)

registerDoParallel(my_cluster)
fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
x <- trClean[,-46]
y <- trClean[,46]
fit <- train(x,y,method = "rf", data=trClean,trControl = fitControl)

confusionMatrix(predict(fit,trClean),trClean[,"classe"])$overall[1]
```

```
## Accuracy
##        1
```

We will now use all three models for prediction on the test data.

LDA:

```
  lda_test_prediction <- predict(ldamod,testClean)
  lda_test_prediction
```

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

RPART:

```
  rpart_prediction <- predict(rpartmod,testClean)
  rpart_prediction
```

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

RandomForrest

```
  rf_prediction <- predict(fit,testClean)
  rf_prediction
```

```
##  [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
```

(model and predict using rf) (explain results) (make a statement about the expected out of sample error)