Prediction Assignment for Practical Machine Learning Coursera

Hsin-Hua Lai

Github address:

https://github.com/hsinhualai/datasciencecoursera/tree/master/Prediction%20Assignment

Overview

In this project, we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, obtained from http://groupware.les.inf.puc-rio.br/har, to predict the manner in which they did the exercise. We first build models based on the training data and use the prediction model to predict 20 different test cases.

Data Cleaning

Before building the predicted models, we first clean the raw training data to eliminate as many predicting variables as possible.

```
## We first set the working repository which contains the raw data files
setwd("/Users/hsinhua/datasciencecoursera/Prediction Assignment")

## First let's import the raw train and test data use read.csv
rawdata_train = read.csv("training.csv",  na.strings = c("","NA", "NULL"))
rawdata_test = read.csv("testing.csv", na.strings = c("","NA", "NULL"))

## Check the dims of the data
dim(rawdata_train)
```

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

```
dim(rawdata_test)
```

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

```
## Let us import the packages that we will use in this project
library(caret)
library(dplyr)
library(gbm)
library(randomForest)
```

If we check the data, we find that lots of the columns contain too many NAs, NULLs, or simply are blank. We can not simply remove the incomplete data rows by using the ‘complete.cases’ command. For example we can first check the percents of the number of NAs in each column using the colSum command

```
napercent <- colSums(is.na(rawdata_train))/nrow(rawdata_train)
```

We do not print the result but we note that lots of columns contain 98% of NAs. Since the ratios of NAs are so high, we conclude the those variables are invalid and we eliminate those columns.

```
process_train <- rawdata_train[,colSums(is.na(rawdata_train)) == 0]
```

Next, let us remove some dummy observables which obviously can not be treated as predictors in regression model, such as "X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window", and "num_window".

```
process2_train <- process_train[,8:length(process_train[1,])]
```

We now check if there are predictors that are near zero variance predictors, which need to be removed since they can not be used as predictors.

```
## We first check which column the zero variance predictor is and
## then remove it
col_nnzv <-  which(nearZeroVar(process2_train, saveMetrics = TRUE)$nzv == FALSE)

process3_train <- process2_train[,col_nnzv]
```

For building models with as fewer predictors as possible, we need to remove the one of the pair variables with very high correlations (greater than 0.9 in this project). Here we use the findCorrelation command in caret package to accomplish this. There are other packages we can use, such as leap and genetic commands in the subselect package, to accomplish the same job.

```
## We first calculate the correlations between each pair of numeric variables
corrM <- cor(process3_train[,sapply(process3_train, is.numeric)])

## Get the variables with high correlations
highcorrvb <- findCorrelation(corrM, cutoff = .9, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.083 vs 0.164 so flagging column 33
## All correlations <= 0.9
```

```r
## Now we remove those variables with very high correlations
process4_train <- process3_train[,-highcorrvb]
```

Building prediction models

Now we split the clean data into training and testing for cross validations.

```r
inTrain <- createDataPartition(process4_train$classe, p = 3/4, list = FALSE)
training <- process4_train[inTrain,]
testing <- process4_train[-inTrain,]
```

Classification Tree Model

We can now build the models below. We first build the classification tree using method rpart in caret package

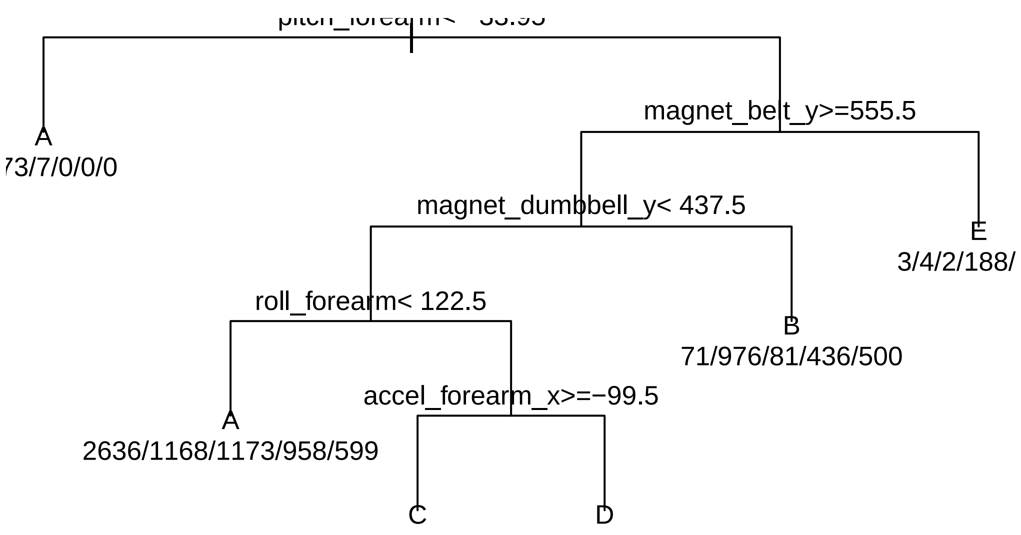
```r
## We use the caret package with method being rpart
modrpart <- train(classe ~., method = "rpart", data = training)

## Print out the final Model
print(modrpart$finalModel)
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) pitch_forearm< -33.95 1180     7 A (0.99 0.0059 0 0 0) *
##    3) pitch_forearm>=-33.95 13538 10526 A (0.22 0.21 0.19 0.18 0.2)
##      6) magnet_belt_y>=555.5 12425  9416 A (0.24 0.23 0.21 0.18 0.14)
##       12) magnet_dumbbell_y< 437.5 10361  7423 A (0.28 0.18 0.24 0.17 0.12)
##         24) roll_forearm< 122.5 6534  3898 A (0.4 0.18 0.18 0.15 0.092) *
##         25) roll_forearm>=122.5 3827  2516 C (0.079 0.18 0.34 0.22 0.18)
##           50) accel_forearm_x>=-99.5 2630  1602 C (0.088 0.22 0.39 0.083 0.22) *
##           51) accel_forearm_x< -99.5 1197   584 D (0.058 0.097 0.24 0.51 0.096) *
##       13) magnet_dumbbell_y>=437.5 2064  1088 B (0.034 0.47 0.039 0.21 0.24) *
##      7) magnet_belt_y< 555.5 1113   197 E (0.0027 0.0036 0.0018 0.17 0.82) *
```

```r
## Plot the classification tree
plot(modrpart$finalModel, uniform = TRUE, main = "Classification Tree")
text(modrpart$finalModel, use.n = TRUE, all = FALSE, cex = 0.8)
```

Classification Tree



We check that he accuracy is 0.4893964, which is very bad.

Generalized Boosted Regreesion Model

Below we try the Generalized Boosted Regression Model (gbm). We do not use the caret package below since it is way too slow.

```r
## We do not use the caret package here since it is way too slow
modgbm <- gbm(classe ~., data = training, distribution = "multinomial",
              n.trees = 200, interaction.depth = 4, shrinkage = 0.005)

## Prediction using the gbm model
predgbm <- predict(modgbm, n.trees = 200, newdata= testing, type = 'response')

## The predict returns back the probability for each classe
## Below for each row we pick the one with largest probability
maxpredgbm <- apply(predgbm, 1, which.max)

## Since 1~5 means A ~ E, we rename them below
maxpredgbm[which(maxpredgbm == 1)] <- "A"
maxpredgbm[which(maxpredgbm == 2)] <- "B"
maxpredgbm[which(maxpredgbm == 3)] <- "C"
maxpredgbm[which(maxpredgbm == 4)] <- "D"
maxpredgbm[which(maxpredgbm == 5)] <- "E"
maxpredgbm <- as.factor(maxpredgbm)
```

The accuracy is 0.771615, which is slightly better than that of classification tree above using caret package.

Random Forest

We below try the random forest model. We again do not use the caret package since it takes forever.

```r
## We again do not use caret package since it takes forever
modrf <- randomForest(classe~., data = training, ntree=100, importance=TRUE, prox = TRUE)

predrf <- predict(modrf, testing)
```

The accuracy is 0.9949021, which is much better than the classification tree model and the generalized boosted model above. According to all the prediction performance above, we use the random forest model to predict 20 different test cases.

Prediction based on the Random Forest Model

The predictions for the 20 different test are B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B.