Machine Learning Final Writeup

Ramon Serres

26 de junio de 2018

Table of Contents

Background

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. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-). You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Cross-validation

Cross-validation will be performed by subsampling our training data into 2 subsamples: sub_Training data (75% of the original Training data set) and sub_Testing data (25% of the original Training data set).

We will build our models will on the sub_Training data set, and tested on the sub_Testing data.

We will pick the best model and it will be finally tested on the original Testing data set.

Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data.

Accuracy : the proportion of correct classified observation over the total sample in the subTesting data set.

Packages loading

Loading required package: lattice Loading required package: ggplot2 Loading required package: caret Loading required package: randomForest Loading required package: rpart Loading required package: rpart.plot

```
suppressMessages(library(lattice))
suppressMessages(library(ggplot2))
suppressMessages(library(caret))

## Warning: package 'caret' was built under R version 3.4.4

suppressMessages(library(randomForest))

## Warning: package 'randomForest' was built under R version 3.4.4

suppressMessages(library(rpart))
suppressMessages(library(rpart.plot))

## Warning: package 'rpart.plot' was built under R version 3.4.4
```

Getting and cleaning data

Data can be found in the following links

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Load data

```
set.seed(2222)

training <- read.csv(trainUrl, na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(testUrl, na.strings=c("NA","#DIV/0!",""))
```

Clean data

We want to remove columns with 100 % empty rows , also remove firts 7 columns from our dataset as those do not contain any relevant information for our prediction purposes

```
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]

training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
```

Cross validation

Use 70% of training set data to built a model, and use the rest to test the model

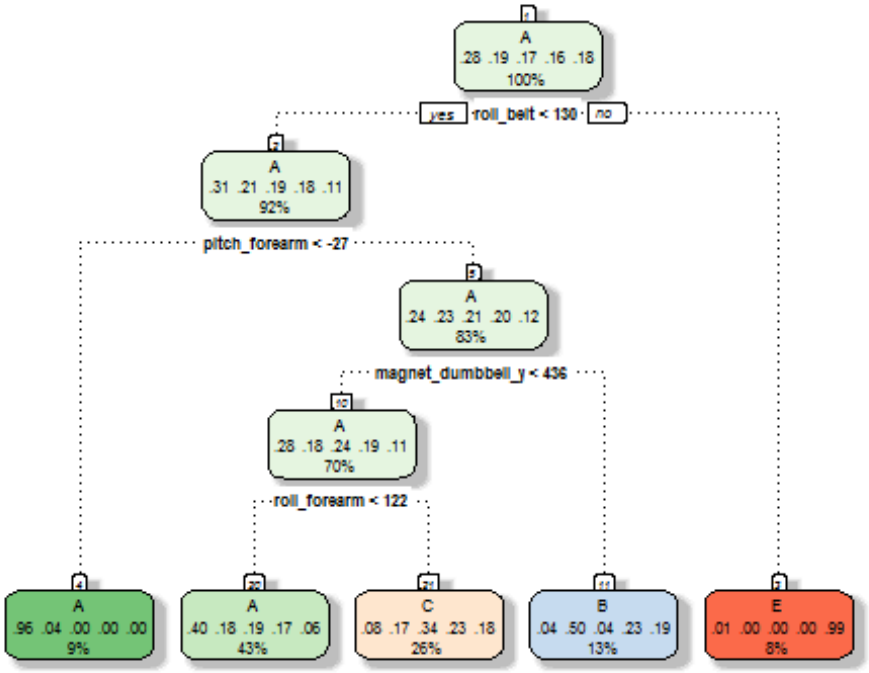
Partitioning training data set in training set into two sub datasets Training (70%) and Testing (30%)

```
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)

sub_Training <- training[inTrain, ]
sub_Testing <- training[-inTrain, ]
```

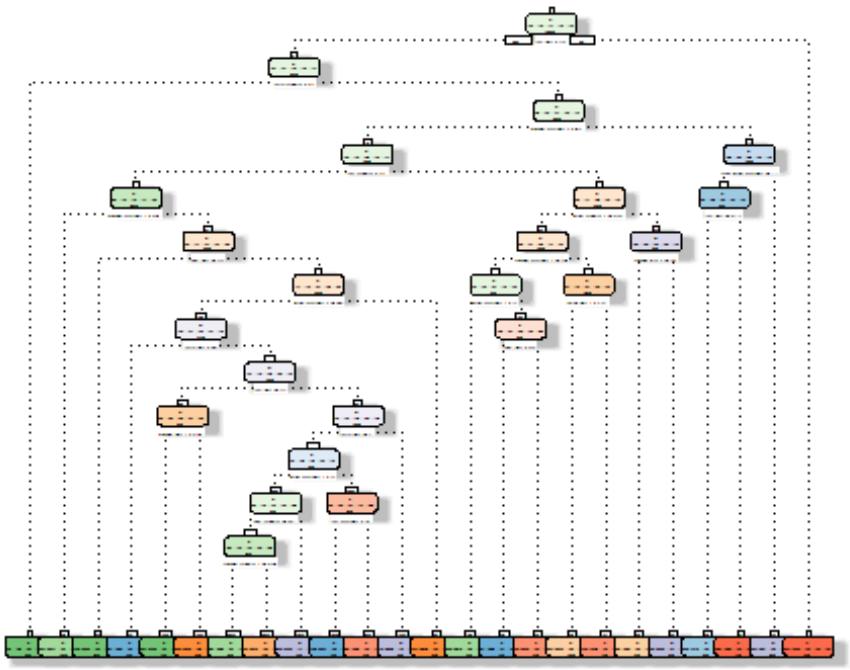
Prediction Option 1: Decisions Tree using train and rpart functions

Let's start by creating a simple Decision Tree so that we have a clean and visual picture of the importance of each feauture

```
suppressMessages(library(rattle))

## Warning: package 'rattle' was built under R version 3.4.4

model_train_rpart<-train(classe ~ .,data = sub_Training,method = "rpart")
model_rpart <- rpart(classe ~ ., data=sub_Training,method="class")


fancyRpartPlot(model_train_rpart$finalModel)
```




```
fancyRpartPlot(model_rpart)

## Warning: labs do not fit even at cex 0.15, there may be some
overplotting
```



See Predicting accuracy with decision trees using both models

```
predictions_model_train_rpart <- predict(model_train_rpart, sub_Testing)
confusionMatrix(predictions_model_train_rpart, sub_Testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1276  396  401  361  133
##          B   21  327   37  144  127
##          C   96  226  417  299  248
##          D    0    0    0    0    0
##          E    2    0    0    0  393
##
## Overall Statistics
##
##                Accuracy : 0.492
##                  95% CI : (0.478, 0.5061)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3357
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9147  0.34457  0.48772   0.0000  0.43618
## Specificity            0.6321  0.91681  0.78538   1.0000  0.99950
## Pos Pred Value         0.4971  0.49848  0.32426      NaN  0.99494
## Neg Pred Value         0.9491  0.85358  0.87894   0.8361  0.88734
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2602  0.06668  0.08503   0.0000  0.08014
## Detection Prevalence   0.5235  0.13377  0.26223   0.0000  0.08055
## Balanced Accuracy      0.7734  0.63069  0.63655   0.5000  0.71784
```

```
predictions_model_rpart <- predict(model_rpart, sub_Testing,type =
"class")
confusionMatrix(predictions_model_rpart, sub_Testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1287  217   30  100   39
##          B   26  488   44   23   57
##          C   29  183  724   99  124
##          D   51   51   56  521   64
##          E    2   10    1   61  617
##
```

```
## Overall Statistics
##
##                Accuracy : 0.7416
##                  95% CI : (0.7291, 0.7538)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6712
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9226  0.51423   0.8468   0.6480   0.6848
## Specificity            0.8900  0.96207   0.8926   0.9459   0.9815
## Pos Pred Value         0.7693  0.76489   0.6247   0.7012   0.8929
## Neg Pred Value         0.9666  0.89194   0.9650   0.9320   0.9326
## Prevalence             0.2845  0.19352   0.1743   0.1639   0.1837
## Detection Rate         0.2624  0.09951   0.1476   0.1062   0.1258
## Detection Prevalence   0.3412  0.13010   0.2363   0.1515   0.1409
## Balanced Accuracy      0.9063  0.73815   0.8697   0.7969   0.8332
```

Predict classe with all the other variables using a random forest ("rf"),boosted trees ("gbm") and linear discriminant analysis ("lda") model.

Stack the predictions together using random forests ("rf")

What is the resulting accuracy on the test set? Is it better or worse than each of the individual predictions?

```
mod_rf <- randomForest(classe ~ .,data = sub_Training,ntree=500)
# mod_gbm <- train(classe ~ .,data = sub_Training, method = "gbm")

# skip boosted trees method as taking too long to compute

mod_lda <- train(classe ~ .,data = sub_Training, method = "lda")

pred_rf <- predict(mod_rf,sub_Testing)

pred_lda <- predict(mod_lda,sub_Testing)
```

Accuracy using random forests

```
confusionMatrix(pred_rf, sub_Testing$classe)$overall[1]

##  Accuracy
## 0.9959217
```

Accuracy using linear discriminant analysis

```
confusionMatrix(pred_lda, sub_Testing$classe)$overall[1]
```

```
##  Accuracy
## 0.7000408
```

Create a staked model

```
predDF <- data.frame(pred_rf, pred_lda,classe=sub_Testing$classe)
combModFit <- train(classe ~ ., method = "rf", data = predDF)

combPred <- predict(combModFit, predDF)
```

Stacked Accuracy

```
confusionMatrix(combPred, sub_Testing$classe)$overall[1]
```

```
##  Accuracy
## 0.9959217
```

Prediction Model to Use:

Random Forest algorithm alone performed better than LDA or Decition Trees .Also Stacked model did not perform any better. So we will choose Ramdom Forest (Accuracy was 0.9961). The expected out-of-sample error is estimated at 0.004, or 0.4%

Predict classes of 20 test data

```
predictfinal <- predict(mod_rf, testing, type="class")
predictfinal
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```