Practical Machine Learning - Course Project

Yannis Moros

July 10, 2017

I have been unable to get GitHub to compile my .html file. Please use this .pdf to grade the project instead. The "raw" .html is still in this repo and you can feel free to compile it using your prefered method, if you wish to. Thanks!

Introduction

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.

The goal of this project is to predict the manner in which they performed the exercise, using the other variables in the dataset. We will train a machine learning model and then use it to predict 20 test cases.

R Setup

```
require(caret)
require(rattle)
require(rpart.plot)
```

Download and read data

```
if(!file.exists("./data")) {
        dir.create("./data")
        url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        file_dest1 <- "./data/pml-training.csv"
        file_dest2 <- "./data/pml-testing.csv"

        download.file(url1, file_dest1)
        download.file(url2, file_dest2)
}

train_raw <- read.csv("./data/pml-training.csv")
test_raw <- read.csv("./data/pml-testing.csv")
```

Data preprocessing

The first step will be to reduce the raw dataset. We will remove the following:

- Variables used for identification purposes
- Variables that consist of more than 90% missing values
- Variables with near zero variation (very few unique values)

```
# Remove id variables
TR <- train_raw[, -c(1:5)]

# Remove variables that are more than 90% missing values
nas <- sapply(TR, function(x) mean(is.na(x))) > 0.9
TR <- TR[, nas == 0]

# Remove variables with almost zero variation (very few unique values)
TR <- TR[, - nearZeroVar(TR)]
```

We would like to have a cross-validation dataset on which we can assess the performance of the model, before predicting the 20 test cases. To achieve that we split the training set as 80% training / 20% validation.

```
inTrain <- createDataPartition(TR$classe, p = 0.8, list = F)
TR <- TR[inTrain, ]
VLD <- TR[-inTrain, ]
```

Now we are ready to try fitting different models.

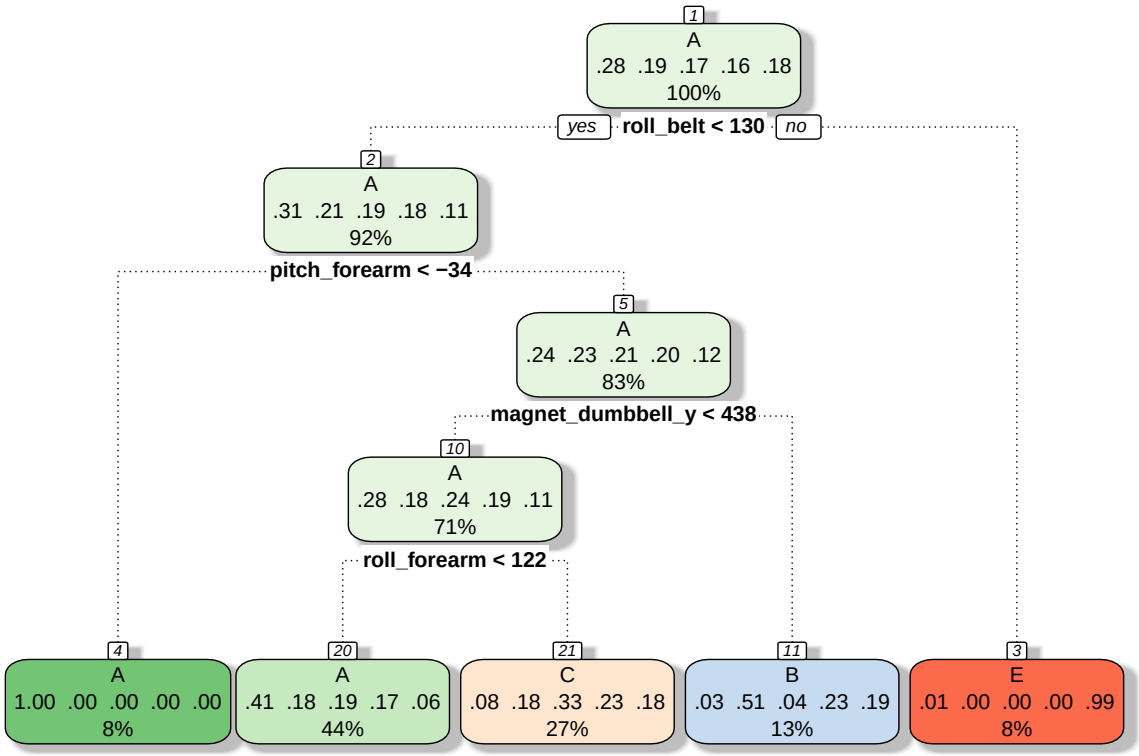
Model 1 - Simple Classification Tree

We will first try a very simple classification tree model.

```
set.seed(666)

# Train the classification tree on the training set
mdl_tree <- train(classe ~ ., method = "rpart", data = TR)

# Create a plot
fancyRpartPlot(mdl_tree$finalModel)
```




```
# Predict the type of exercise on the validation set
pred_tree <- predict(mdl_tree, VLD)

# View the performance of the model and assess the accuracy / out-of-sample error.
confusionMatrix(VLD$classe, pred_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 801  15  73   0   3
##          B 238 215 159   0   0
##          C 251  14 266   0   0
##          D 235  84 207   0   0
##          E  88  79 178   0 241
##
## Overall Statistics
##
##                Accuracy : 0.484
##                  95% CI : (0.4664, 0.5016)
##     No Information Rate : 0.5126
##     P-Value [Acc > NIR] : 0.9994
##
##                   Kappa : 0.3279
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4966  0.52826  0.30125       NA  0.98770
## Specificity            0.9407  0.85511  0.88295   0.8329  0.88116
## Pos Pred Value         0.8980  0.35131  0.50094       NA  0.41126
## Neg Pred Value         0.6399  0.92426  0.76414       NA  0.99883
## Prevalence             0.5126  0.12933  0.28058   0.0000  0.07753
## Detection Rate         0.2545  0.06832  0.08452   0.0000  0.07658
## Detection Prevalence   0.2834  0.19447  0.16873   0.1671  0.18621
## Balanced Accuracy      0.7186  0.69168  0.59210       NA  0.93443
```

With just 49% accuracy, or 51% expected out-of-sample error, this simple model is not good enough to predict the test cases.

Model 2 - Random Forest

Now we will try fitting a simple Random Forest model.

```
set.seed(13)
# Train the model on the training set
mdl_for <- train(classe ~ ., method = "rf", data = TR)

# View the final model
print(mdl_for)
```

```
## Random Forest
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9927108  0.9907762
##   27    0.9964551  0.9955142
##   53    0.9921042  0.9900092
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Predict on the validation set
pred_for <- predict(mdl_for, VLD)

# Assess the performance and view the confusion matrix
confusionMatrix(VLD$classe, pred_for)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
```

```
##          A 892   0   0   0   0
##          B   0 612   0   0   0
##          C   0   0 531   0   0
##          D   0   0   0 526   0
##          E   0   0   0   0 586
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2834
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2834   0.1945   0.1687   0.1671   0.1862
## Detection Rate         0.2834   0.1945   0.1687   0.1671   0.1862
## Detection Prevalence   0.2834   0.1945   0.1687   0.1671   0.1862
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The accuracy of this random forest model is actually 100% (0 out-of-sample error), as it predicted correctly every single exercise in the validation test.

This hints to overfitting, but we will still use this to predict the type of exercise in the 20 test cases.

```
pred_quiz <- predict(mdl_for, test_raw)
pred_quiz
```

```
##  [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
```