Coursera - Practical Machine Learning (Prediction Assignment Writeup)

Anil Kumar

February 28, 2016

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. 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).

The project can be found on github: http://benakiva.github.io/practicalML/ and https://github.com/benakiva/practicalMLu can also embed plots, for example:

Source of 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.

Loading Data

In this section, we download the data files from the Internet and load them into two data frames. We ended up with a training dataset and a 20 observations testing dataset that will be submitted to Coursera.

```
# Install package if not installed
# install.packages("caret")
# install.packages("rattle")
# install.packages("rpart.plot")
# install.packages("randomForest")
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

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

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

```
# Load the training dataset
dt_training <- read.csv("C:/temp/pml-training.csv", na.strings=c("NA",""),
strip.white=T)

# Load the testing dataset
dt_testing <- read.csv("C:/temp/pml-testing.csv", na.strings=c("NA",""),
strip.white=T)
```

Data Cleansing

In this section, we will remove all columns that contains NA and remove features that are not in the testing dataset. The features containing NA are the variance, mean and standard devition (SD) within each window for each feature. Since the testing dataset has no time-dependence, these values are useless and can be disregarded. We will also remove the first 7 features since they are related to the time-series or are not numeric.

```
features <- names(dt_testing[,colSums(is.na(dt_testing)) == 0])[8:59]

# Only use features used in testing cases.
dt_training <- dt_training[,c(features,"classe")]
dt_testing <- dt_testing[,c(features,"problem_id")]
```

```
dim(dt_training); dim(dt_testing);

## [1] 19622    53

## [1] 20 53
```

Partitioning the Dataset

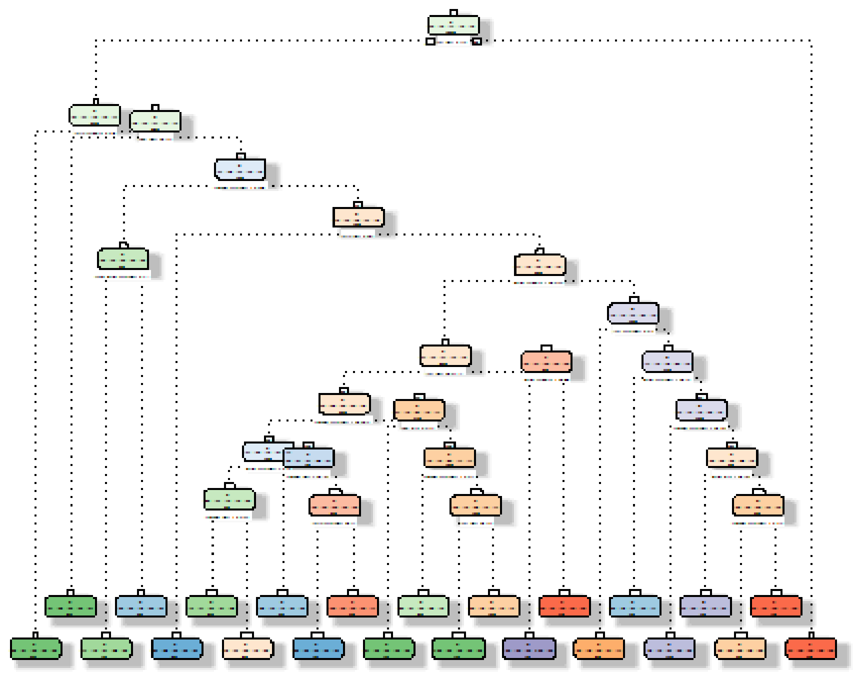
Following the recommendation in the course Practical Machine Learning, we will split our data into a training data set (60% of the total cases) and a testing data set (40% of the total cases; the latter should not be confused with the data in the pml-testing.csv file). This will allow us to estimate the out of sample error of our predictor.

```
set.seed(12345)

inTrain <- createDataPartition(dt_training$classe, p=0.6, list=FALSE)
training <- dt_training[inTrain,]
testing <- dt_training[-inTrain,]

dim(training); dim(testing);

## [1] 11776    53

## [1] 7846   53
```

Building the Decision Tree Model

Using Decision Tree, we shouldn't expect the accuracy to be high. In fact, anything around 80% would be acceptable.

```
modFitDT <- rpart(classe ~ ., data = training, method="class")
fancyRpartPlot(modFitDT)

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


Rattle 2016-Feb-28 10:40:43 Anil Kumar

Predicting with the Decision Tree Model

```
set.seed(12345)

prediction <- predict(modFitDT, testing, type = "class")
confusionMatrix(prediction, testing$class)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##            A 1879  260   30   69   66
##            B   56  759   88   34   54
##            C  105  340 1226  354  234
##            D  155  132   23  807   57
##            E   37   27    1   22 1031
##
## Overall Statistics
##
##                Accuracy : 0.7267
##                  95% CI : (0.7167, 0.7366)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6546
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8418  0.50000   0.8962   0.6275   0.7150
## Specificity            0.9243  0.96334   0.8405   0.9441   0.9864
## Pos Pred Value         0.8155  0.76589   0.5427   0.6874   0.9222
## Neg Pred Value         0.9363  0.88928   0.9746   0.9282   0.9389
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2395  0.09674   0.1563   0.1029   0.1314
## Detection Prevalence   0.2937  0.12631   0.2879   0.1496   0.1425
## Balanced Accuracy      0.8831  0.73167   0.8684   0.7858   0.8507
```

Building the Random Forest Model

Using random forest, the out of sample error should be small. The error will be estimated using the 40% testing sample. We should expect an error estimate of < 3%.

```
set.seed(12345)
modFitRF <- randomForest(classe ~ ., data = training, ntree = 1000)
```

Predicting on the Testing Data (pml-testing.csv)

```
predictionDT <- predict(modFitDT, dt_testing, type = "class")
predictionDT

##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  C  A  C  A  A  E  D  D  A  A  A  C  A  A  C  E  A  D  C  B
## Levels: A B C D E
```

Random Forest Prediction

```
predictionRF <- predict(modFitRF, dt_testing, type = "class")
predictionRF

##  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
```

Conclusion

As can be seen from the confusion matrix the Random Forest model is very accurate, about 99%. Because of that we could expect nearly all of the submitted test cases to be correct. It turned out they were all correct.