Week 4

Practical Machine Learning

Prediction Assignment Writeup

Introduction

Using devices like Jawbone Up, Nike FuelBand, and Fitbit it's now possible to gather an oversized amount of information about personal activity relatively inexpensively. These form of devices are a part of the quantified self movement - a bunch of enthusiasts who take measurements about themselves regularly to boost their health, to search out patterns in their behavior, or because they're tech geeks. One thing that folks regularly do is quantify what proportion of a selected activity they are doing, but they rarely quantify how well they are doing it. during this project, your goal are 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 other ways.

Data Source

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 the Dataset

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

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3
.3
```

```
library(RColorBrewer)
library(RGtk2)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.3
## Rattle: A free graphical interface for data science with R
.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3
.3.3
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:rattle':
##
##     importance
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-trai
ning.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv"


# download the datasets
dt_training <- read.csv(url(UrlTrain))
dt_testing  <- read.csv(url(UrlTest))
```

Cleaning the Data

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

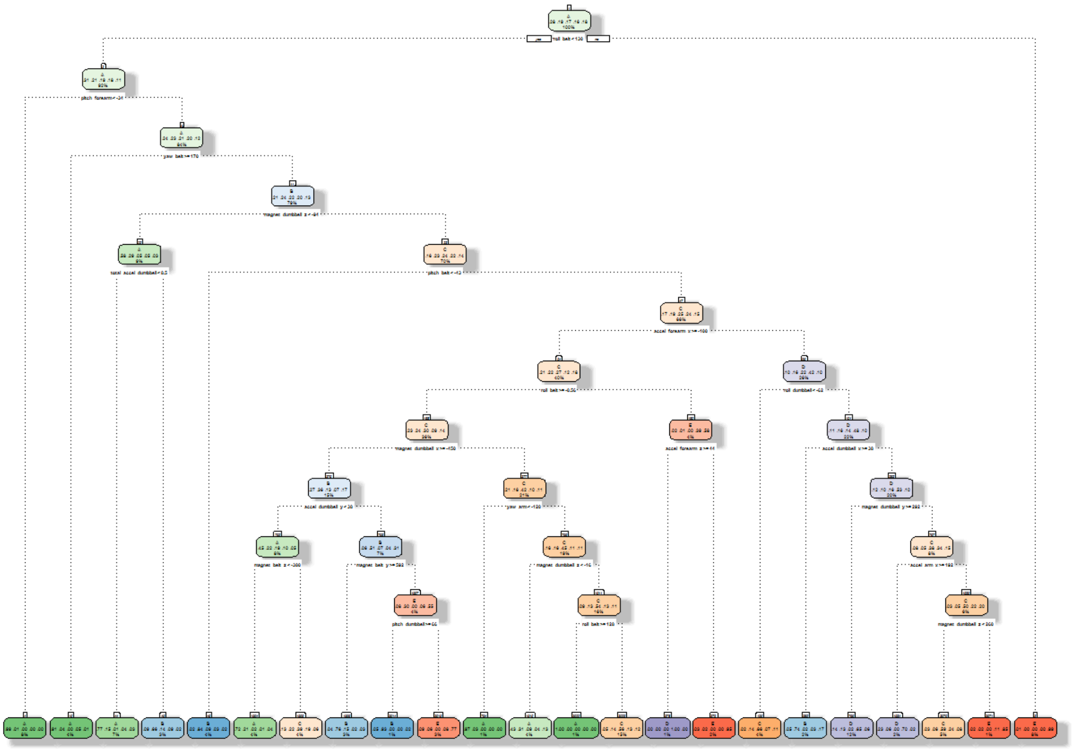
Following the recommendation in the course Practical Machine Learning, we will split our data into a training data set (60% of the total cases) and a testing data set (40% of the total cases; the latter should not be confused with the data in the pml-testing.csv file). This will allow us to estimate 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)
## [1] 11776    53
dim(testing)
## [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)
```



Rattle 2017-Dec-03 09:37:30 Amit

Predicting with the Decision Tree Model

```
set.seed(12345)


prediction <- predict(modFitDT, testing, type = "class")
confusionMatrix(prediction, testing$classe)
## 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    0.17
44    0.1639    0.1838
## Detection Rate                   0.2395   0.09674    0.1563
  0.1029    0.1314
## Detection Prevalence    0.2937   0.12631    0.2879    0.14
96    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 with the Random Forest Model

```
prediction <- predict(modFitRF, testing, type = "class")
confusionMatrix(prediction, testing$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    9    0    0
0
##          B    2 1504    6    0
0
##          C    0    5 1362   17    2
##          D    0    0    0 1267
5
##          E    0    0    0    2 143
5
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9919, 0.9955
)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class:
B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9908   0.99
56   0.9852   0.9951
```

```
## Specificity                          0.9984    0.9987    0.99
63     0.9992    0.9997

## Pos Pred Value                      0.9960    0.9947    0.9827
    0.9961    0.9986

## Neg Pred Value                      0.9996    0.9978    0.9991
    0.9971    0.9989

## Prevalence                           0.2845    0.1935    0.1
744     0.1639    0.1838

## Detection Rate                     0.2842    0.1917    0.1736
    0.1615    0.1829

## Detection Prevalence       0.2854    0.1927    0.1767    0.
1621     0.1832

## Balanced Accuracy               0.9988    0.9948    0.9960
 0.9922    0.9974
```

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

Decision Tree Prediction

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

```
##    1    2    3    4    5    6    7    8    9  10  11  12  13  1
4  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  1
4  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
```

Submission file

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.

Prepare the submission.

```
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}


pml_write_files(predictionRF)
```