Practical Machine Learning Final

Phil D.

4/16/2017

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
library(RColorBrewer)
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
```

```
library(knitr)
library(rattle)
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```
## Please install GTK+ from http://r.research.att.com/libs/GTK_2.24.17-X11.pkg
```

```
## If the package still does not load, please ensure that GTK+ is installed and that it is on your PATH
```

```
## IN ANY CASE, RESTART R BEFORE TRYING TO LOAD THE PACKAGE AGAIN
```

```
## 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.
```

```
set.seed(16161)
```

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).

Getting and Cleaning Data

Importing the data:

```
training <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), na.str
testing <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), na.strin
```

Split the training data into a training and test set (for model refinement purposes).

```
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
```

Clean the data by removing near zero variables and columns that contain N/As. This data would expand the complexity of the model without providing additional value.

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]
myTesting <- myTesting[,nzv$nzv==FALSE]
testing <- testing[,nzv$nzv==FALSE]

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

The first seven columns of data are purely describing the experiment. They are not relevant to the prediction. I remove them to maintain simplicity in my model.

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

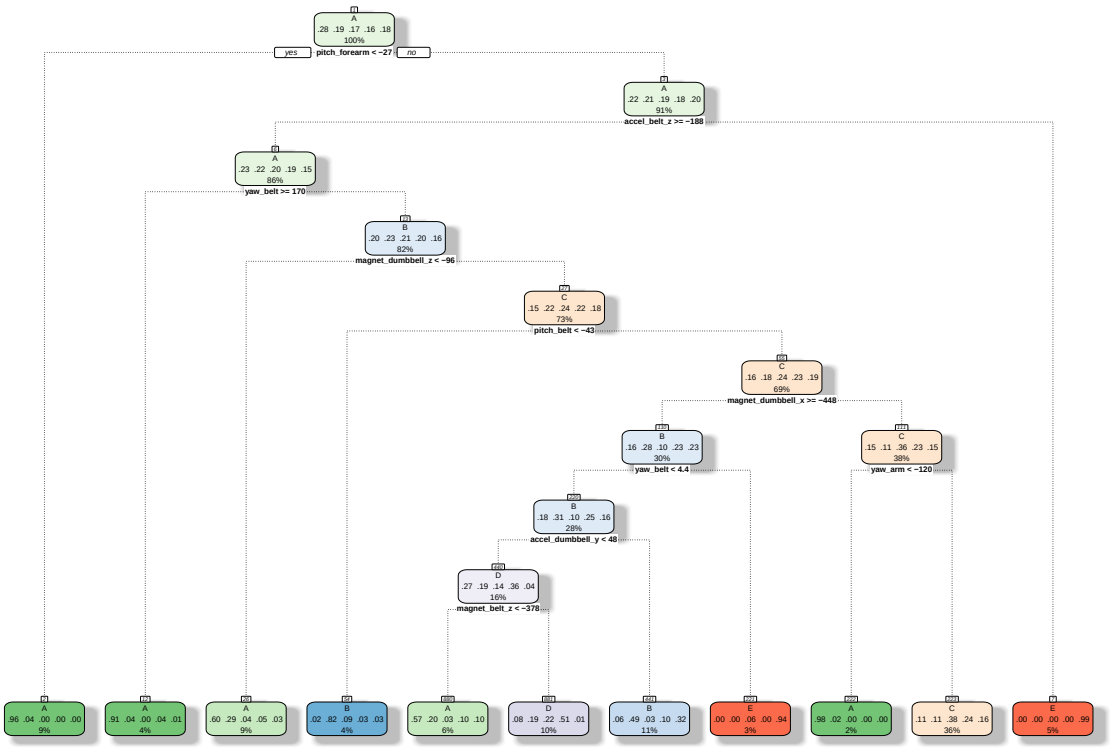
Classification Tree Model

Creating the classification tree model

```
control <- trainControl(method = "cv", number = 5)
fit_rpart <- train(classe ~ ., data = myTraining, method = "rpart",
                   trControl = control)
```

Visual of the tree

```
fancyRpartPlot(fit_rpart$finalModel)
```



Rattle 2017−Apr−16 21:33:30 pdrexler

```
pred<-predict(fit_rpart,myTesting)
conf_rpart <- confusionMatrix(myTesting$classe, pred)
conf_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1811   48  312   58    3
##          B  368  691  318  140    1
##          C   61   52 1084  160   11
##          D  117   87  672  410    0
##          E   70  319  442    6  605
##
## Overall Statistics
##
##                Accuracy : 0.5864
##                  95% CI : (0.5754, 0.5973)
##     No Information Rate : 0.3604
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4758
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7462  0.57728   0.3833  0.52972  0.97581
## Specificity            0.9223  0.87562   0.9434  0.87613  0.88417
```

```
## Pos Pred Value         0.8114  0.45520   0.7924  0.31882  0.41956
## Neg Pred Value         0.8903  0.92004   0.7308  0.94451  0.99766
## Prevalence             0.3093  0.15256   0.3604  0.09865  0.07902
## Detection Rate         0.2308  0.08807   0.1382  0.05226  0.07711
## Detection Prevalence   0.2845  0.19347   0.1744  0.16391  0.18379
## Balanced Accuracy      0.8342  0.72645   0.6634  0.70292  0.92999
```

Using model to predict the test data indicates that the accuracy of the model is only ~58%.

Random Forest Model

In an attempt to get a more accurate model, I used the random forest method.

```
fit_rf <- train(classe ~ ., data = myTraining, method = "rf",
                trControl = control)
```

```
pred<-predict(fit_rf,myTesting)
conf_rf <- confusionMatrix(myTesting$classe, pred)
conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    3    0    0    3
##          B   21 1494    3    0    0
##          C    0   12 1343   13    0
##          D    0    0   15 1270    1
##          E    0    0    1    5 1436
##
## Overall Statistics
##
##                Accuracy : 0.9902
##                  95% CI : (0.9877, 0.9922)
##     No Information Rate : 0.2864
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9876
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9907   0.9901   0.9860   0.9860   0.9972
## Specificity            0.9989   0.9962   0.9961   0.9976   0.9991
## Pos Pred Value         0.9973   0.9842   0.9817   0.9876   0.9958
## Neg Pred Value         0.9963   0.9976   0.9971   0.9973   0.9994
## Prevalence             0.2864   0.1923   0.1736   0.1642   0.1835
## Detection Rate         0.2837   0.1904   0.1712   0.1619   0.1830
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9948   0.9931   0.9911   0.9918   0.9981
```

This time predicting based on my test set I find an accuracy of 99%.

Predicting Final Test Set

I use this random forest model to predict the quiz answers and got the correct answers.

```
predict(fit_rf,testing)
```

```
##  [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
```