Course Project PML

Ryan Kramer

9/18/2017

Background Information

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 purpose of this project is to find a model that can effectively predict which class of workouts each participant performs.

Data Sources

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 comes from this original source: http://groupware.les.inf.puc-rio.br/har.

Load Necessary Libraries

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
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(e1071)
```

Loading the Data

Store URL locations of files as variables.

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

Create training and testing data frames:

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

Partitioning Training Set

60% of the training set will be used for training, while 40% will be stored as a testing set.

```
set.seed(34)
Train <- createDataPartition(y=training$classe, p =0.6, list = FALSE)
myTrain <- training[Train, ]
myTest <- training[-Train, ]
```

Cleaning the Data

Step 1: Cleaning Near Zero Variance variables.

```
nzvTrain <- nearZeroVar(myTrain, saveMetrics=TRUE)

nzvVars <- names(myTrain) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")

myTrain <- myTrain[!nzvVars]
```

Step 2: Removing first column.

```
myTrain <- myTrain[c(-1)]
```

Step 3: Cleaning variables wth too man NAs. The threshold for NAs will be 60%.

```
trainingStep3 <- myTrain
for(i in 1:length(myTrain)) {
        if( sum( is.na( myTrain[, i] ) ) /nrow(myTrain) >= .6 ) {
        for(j in 1:length(trainingStep3)) {
            if(length( grep(names(myTrain[i]), names(trainingStep3)[j]) ) ==1)  {
                trainingStep3 <- trainingStep3[ , -j]
            }
        }
    }
}

myTrain <- trainingStep3
rm(trainingStep3)
```

Now, the test data must be cleaned.

```
clean1 <- colnames(myTrain)
clean2 <- colnames(myTrain[, -58]) #already with classe column removed
myTest <- myTest[clean1]
testing <- testing[clean2]
```

Lastly, the data must be coerced into the same type for Decision Trees and Random Forest algorithms to work correctly.

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTrain)) {
        if(length(grep(names(myTrain[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTrain[i])
        }
    }
}
testing <- rbind(myTrain[2, -58] , testing)
testing <- testing[-1,]
```

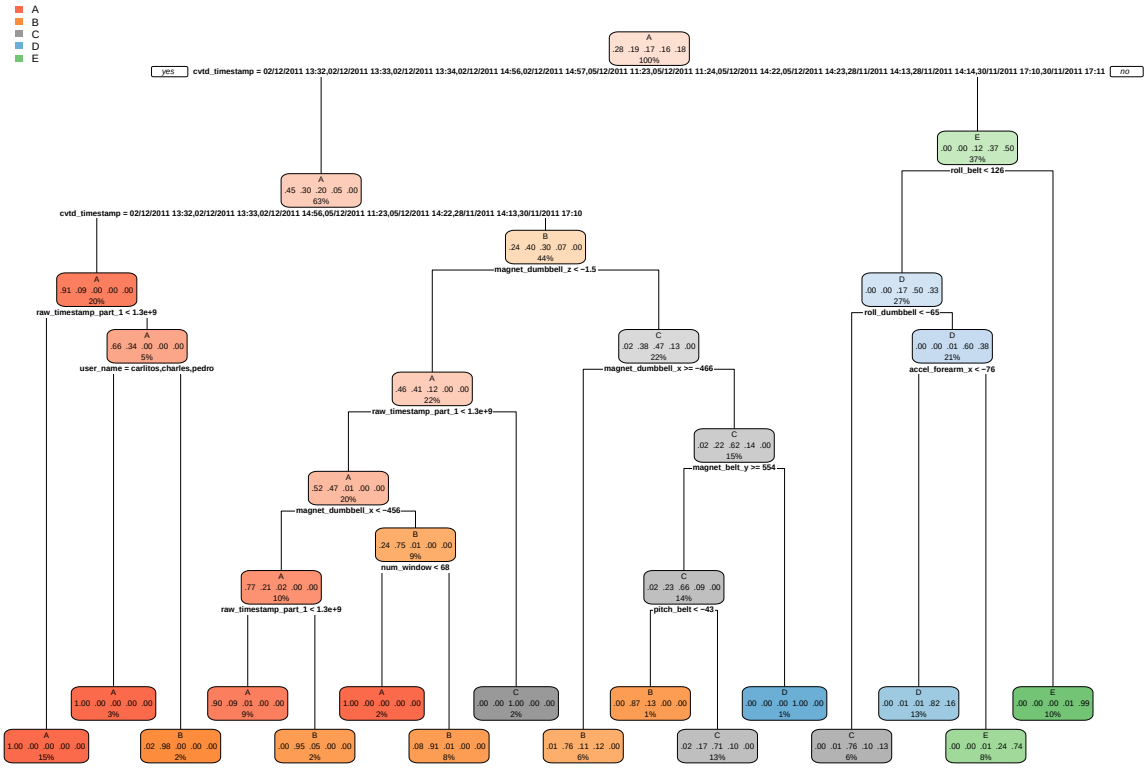
Decision Tree

To create a decision tree fit:

```
dtFit<- rpart(classe~., data = myTrain, method = "class")
```

To view the decision tree:

```
rpart.plot(dtFit)
```



Now, this model will be used to predict the testing set that was set aside from the training observations.

```
predicted <- predict(dtFit, myTest, type = "class")
```

Confusion matrix will be created to examine the results:

```
confusionMatrix(predicted, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   66    7    4    0
##          B   62 1268   85   66    0
##          C   20  173 1253  141   57
##          D    0   11   10  883  201
##          E    0    0   13  192 1184
##
## Overall Statistics
##
##                Accuracy : 0.8588
##                  95% CI : (0.8509, 0.8664)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8214
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9633   0.8353   0.9159   0.6866   0.8211
## Specificity            0.9863   0.9663   0.9396   0.9662   0.9680
## Pos Pred Value         0.9654   0.8562   0.7622   0.7991   0.8524
## Neg Pred Value         0.9854   0.9607   0.9815   0.9402   0.9600
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1616   0.1597   0.1125   0.1509
## Detection Prevalence   0.2838   0.1888   0.2095   0.1408   0.1770
## Balanced Accuracy      0.9748   0.9008   0.9278   0.8264   0.8945
```

Random Forests

The random forest is created in the following code chunk.

```
rfFit <- randomForest(classe~., data = myTrain)
```

Predicting the test observations held out of training set:

```
predictedRF <- predict(rfFit, myTest, type = "class")
```

Confusion matrix will be created to examine the results:

```
confusionMatrix(predictedRF, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    2    0    0
##          C    0    0 1366    3    0
##          D    0    0    0 1281    1
##          E    0    0    0    2 1441
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9985   0.9961   0.9993
## Specificity            0.9998   0.9997   0.9995   0.9998   0.9997
## Pos Pred Value         0.9996   0.9987   0.9978   0.9992   0.9986
## Neg Pred Value         1.0000   0.9998   0.9997   0.9992   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1741   0.1633   0.1837
## Detection Prevalence   0.2846   0.1936   0.1745   0.1634   0.1839
## Balanced Accuracy      0.9999   0.9995   0.9990   0.9980   0.9995
```

The random forest yields better results than the decision tree.