Practical Machine Learning - Course Project

Ryan Wood

March 31, 2017

Background

This is the course project for Practical Machine Learning. The objective is to use personal activity data from acceleromters to predict what type of exercise indiviudals are performing. The data source is available at http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). This analysis attempted fitting 3 types of predictive models to the dataset; Random Forest, Recursive Partitioning, and Linear Discriminant Analysis. The model with the best accuracy was chosen as the preferred model.

Data

There are 160 variables including the classe variable, the manner in which the exercise was performed, which this model aims to predict. The training dataset has 19622 cases and the final testing dataset has 20 cases.

Cleaning the Data

The dataset has several variables which are not relevant and should be removed to avoid impacting the algorythms (Columns 1 through 7 have ID's, timestamps, periods, etc). NA values should also be identified so they can be uniformly removed.

```
library(data.table)
training <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', na.strings=
testing <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv', na.strings=c
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]
nrow(training)
```

```
## [1] 19622
```

```
nrow(testing)
```

```
## [1] 20
```

```
ncol(training)
```

```
## [1] 153
```

```
ncol(testing)
```

```
## [1] 153
```

Cross Validation Planning

We begin by partitioning the training file into a training subset and testing subset. This will allow for cross validation where we will have the ability to measure the accuracy of a model's prediction using an independent test set. This partition is random in order to attempt to have an unbiased testing set. We can use the r package "caret" to automate this partitioning.

We should also remove NA values to help clean the data. If a variable is 60% NA it was removed from the analysis. This leaves us with 52 possilble predictor variables plus the the variable "classe".

```
library(caret)
```

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

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

```
cv <- createDataPartition(y=training$classe, p=0.70, list = FALSE)
cvTraining <- training[cv, ]
cvTesting <- training[-cv, ]

navalues <- colSums(is.na(cvTraining))/nrow(cvTraining)
nanames <- names(navalues[navalues > .6])
navariables <- which(navalues > .6)

cleancvTraining <- cvTraining[,-navariables]

variables <- names(cleancvTraining)
variables
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

Model Development

We can attempt several differnet model algorthyms to find a version with high accruacy and low error rate.

1. Recursive Partioning Model has low accuracy of 49% in the training model.

```
Rpartmod <- train(classe ~ .,data = cleancvTraining, method = "rpart")
```

```
## Loading required package: rpart
```

```
predRpart <- predict(Rpartmod, cvTraining)
confusionMatrix(predRpart, cvTraining$classe)$overall[1]
```

```
##  Accuracy
## 0.4950135
```

2. LDA model achieved 70% accuracy in the training model.

```
Lmod <- train(classe ~ .,data = cleancvTraining, method = "lda")
```

```
## Loading required package: MASS
```

```
predLmod <- predict(Lmod, cvTraining)
confusionMatrix(predLmod, cvTraining$classe)$overall[1]
```

```
##  Accuracy
## 0.7091068
```

3. Random Forest Model achieved 100% accuracy in the training model and 99% accuracy in the testing model.

```
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
```

```
set.seed(5555)
RFmod <- randomForest(classe ~., data=cleancvTraining, type="class")
pred_RFmodTrain <- predict(RFmod, cvTraining)
confusionMatrix(pred_RFmodTrain, cvTraining$classe)$overall[1]
```

```
## Accuracy
##        1
```

Figure 1: Random Forest Model Training Sample Confusion Matrix

```
confusionMatrix(pred_RFmodTrain, cvTraining$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
```

Random Forest model has 100% Accuracy with Training Data, so the next step is to see how this model works for the Testing Data.

```
pred_RFmodTest <- predict(RFmod, cvTesting)
confusionMatrix(pred_RFmodTest, cvTesting$classe)$overall[1]
```

```
## Accuracy
## 0.995582
```

Expected out-of-sample error: 99% Accuracy with Testing Data, so this model is the preferred choice to use on the Test dataset.

Random Forest Model Predictions on the Test Set

```
Test <- predict(RFmod, testing, type="class")
Test
```

```
##  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
```