Analysis of Weight Lifting Exercise

Dataset and initial work

Prior to data analysis, the complete dataset was loaded and summarized. Many of the variables had a large number of missing values, so the data was subset to include only the variables with complete cases. This resulted in 52 predictors, plus the predicted variable of classe.

```
setwd("/Users/bim/Documents/Personal/Coursera/Johns Hopkins Data Science 2014/Practical Machine Learning
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2

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

```
library(rpart)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.1

## Rattle: A free graphical interface for data mining with R.
## Version 3.3.0 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.1.2
```

```
keep <- c(8:11,37:49,60:68,84:86,102,113:124,140,151:160)
trainImport <- read.csv("pml-training.csv")[keep]
testImport <- read.csv("pml-testing.csv")[keep]
```

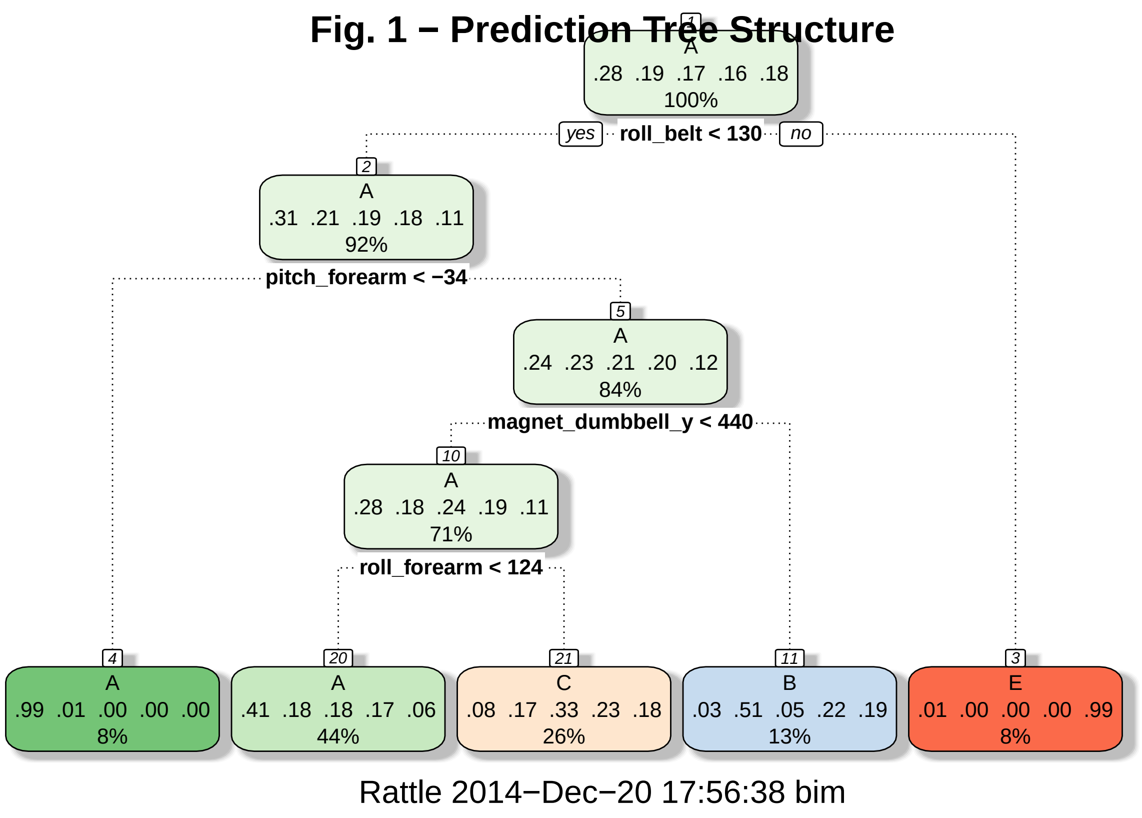
Learning Algorithm

A few different learning mechanisms were explored, including bagging and random forests. However, the dataset was large enough (at least on my computer) to make the time to run prohibitive. Additionally, I had concerns about overfitting and the ability to cross-validate. Therefore, I decided to use a prediction tree. The training set was split into two subsets, with one of them (70%) used as training and the other (30%) as testing of the model. The training set was then fit with a tree from the caret package.

```
inTrain <- createDataPartition(y=trainImport$classe, p=0.7, list=FALSE)
training <- trainImport[inTrain,]
testing <- trainImport[-inTrain,]
fitTree <- train(classe ~ ., data=training, method="rpart")
```

The resulting tree is shown in Fig. 1.

```
fancyRpartPlot(fitTree$finalModel, main="Fig. 1 - Prediction Tree Structure")
```



Performance In order to assess performance, model predictions were made for each subset (training and testing). Then, I wrote a couple of functions to analyze the predictions. First, I looked at which outcome was predicted with the highest probability, and compared that to the actual case.

```
trainPred <- predict(fitTree$finalModel, newdata=training)
testPred <- predict(fitTree$finalModel, newdata=testing)

getPrediction <- function(predMatrix, truth) {
        classes <- c("A","B","C","D","E")
        pmax <- apply(predMatrix, 1, max)
        pcol <- apply(predMatrix, 1, which.max)
        predClass <- classes[pcol]
        x <- data.frame(predClass, pmax, truth)
        names(x) <- c("Predict", "Prob", "Truth")
        x$Correct <- as.character(x$Predict)==as.character(x$Truth)
        return(x)
}

trainingResults <- getPrediction(trainPred, training$classe)
testResults <- getPrediction(testPred, testing$classe)
percCorrTrain <- sum(trainingResults$Correct)/nrow(trainingResults)
percCorrTest <- sum(testResults$Correct)/nrow(testResults)
```

The resulting matrix for the test subset is shown in Fig. 2. The diagonals correspond to correct responses. Note that detection of outcome A (which is the “proper” lifting method) is better than the other outcomes.

Overall, the percent correct prediction was 0.4948. Since there are 5 possible outcomes, this is better than chance guessing, which would be .20, but it is still not great performance.

Fig. 2 - Prediction matrix for test subset

```r
table(testResults$Predict, testResults$Truth)
```

```
##
##          A    B    C    D    E
##     A 1524  465  480  443  150
##     B   26  379   26  173  139
##     C  120  295  520  348  304
##     E    4    0    0    0  489
```

Next, I looked at the probability associated with what the correct answer actually was, regardless of whether that was the highest probability or not, and used that probability as a continuous variable in order to calculate RMS error.

```r
getProb <- function(probs,targets) {
        values <- vector()
        for(i in c(1:length(targets))) {
                values <- c(values, probs[i,targets[i]])
        }
        return(values)
}

correctTraining <- getProb(as.data.frame(trainPred),training$classe)
rmsErrorTrain <- sqrt(sum((1-correctTraining)^2)/length(correctTraining))
correctTesting <- getProb(as.data.frame(testPred),testing$classe)
rmsErrorTest <- sqrt(sum((1-correctTesting)^2)/length(correctTesting))
```

For the training subset, the RMS error was 0.6804, and for the test subset it was 0.6808. Although the error rate is fairly high (similar to the observation above), one positive note is that the rates are not substantially different from training set to test set. This indicates that the data has not been overfit.

Cross-Validation

One potential issue with prediction trees is that they can produce variable results. In order to control for this and estimate the out-of-sample error better, I wrote a function to cross-validate by randomly subsetting the data 10 different times. That way, the average of all error estimates can be taken.

```r
numreps <- 10
allRMStrain <- vector()
allRMStest <- vector()
for(i in c(1:numreps)) {
        inTrain <- createDataPartition(y=trainImport$classe, p=0.7, list=FALSE)
        training <- trainImport[inTrain,]
        testing <- trainImport[-inTrain,]

        fitTree <- train(classe ~ ., data=training, method="rpart")
        trainPred <- predict(fitTree$finalModel, newdata=training)
        testPred <- predict(fitTree$finalModel, newdata=testing)
        trainingResults <- getPrediction(trainPred, training$classe)
        testResults <- getPrediction(testPred, testing$classe)

        correctTraining <- getProb(as.data.frame(trainPred),training$classe)
        rmsErrorTrain <- sqrt(sum((1-correctTraining)^2)/length(correctTraining))
        correctTesting <- getProb(as.data.frame(testPred),testing$classe)
        rmsErrorTest <- sqrt(sum((1-correctTesting)^2)/length(correctTesting))

        allRMStrain <- c(allRMStrain, rmsErrorTrain)
        allRMStest <- c(allRMStest, rmsErrorTest)
}

overallRMStrain <- mean(allRMStrain)
overallRMStest <- mean(allRMStest)
```

A plot of the RMS values for each of the test sets is shown in Fig. 3. The average of all RMS errors for the training sets is 0.6789, and for the test sets it is 0.6787. This value (0.6787) is my best estimate of the out-of-sample error rate.

```
plot(allRMStest, xlab="Iteration", ylab="RMS Error", main="Fig. 3. - Error Rates Across Multiple Test S
```

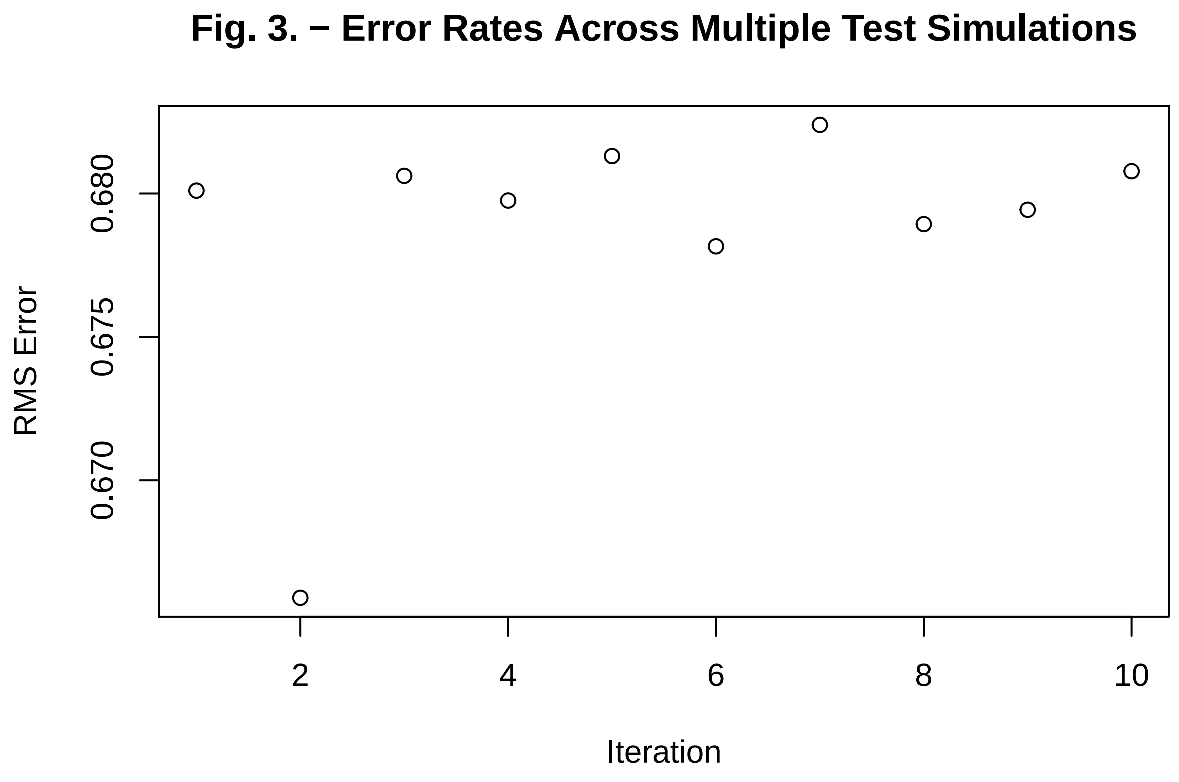


Fig. 3. − Error Rates Across Multiple Test Simulations
RMS Error
Iteration