Classifying Movement

Our goal is to use machine learning techniques to build a model to predict the "classe" category that a particular exercise belongs to based on a set of measurements from the exercise.

First we read in the data which has been pre-classified. Looking at the summary we can see that most of the variables are over 99% missing, so we exclude those variables to reduce the chance of overfitting based on their presence and to allow using techniques that don't handle missing values well.

```
library(caret)
```

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

```
library(rpart)
library(plyr)

setwd("~/Code/datasciencecoursera/Machine learning/")
alltrain <- read.csv("pml-training.csv")
#summary(alltrain)
keeptrain <- alltrain[c("roll_belt", "pitch_belt",  "yaw_belt", "total_accel_belt", "gyros_belt_x", "gy
    "accel_belt_x", "accel_belt_y", "accel_belt_z", "magnet_belt_x", "magnet_belt_y", "magnet_belt_z",
    "roll_arm", "pitch_arm", "yaw_arm", "total_accel_arm", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z",
    "accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x", "magnet_arm_y", "magnet_arm_z",
    "roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell", "total_accel_dumbbell", "gyros_dumbbell_x", "gyro
    "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "magnet_dumbbell_x", "magnet_dumbbell_y
    "roll_forearm", "pitch_forearm", "yaw_forearm", "total_accel_forearm", "gyros_forearm_x", "gyros_fo
    "accel_forearm_x", "accel_forearm_y", "accel_forearm_z", "magnet_forearm_x", "magnet_forearm_y", "ma
```

We split the data into training and testing data sets so we can verify our model is not overfitting. Otherwise we would likely have small in-sample errors but very large out-of-sample errors.

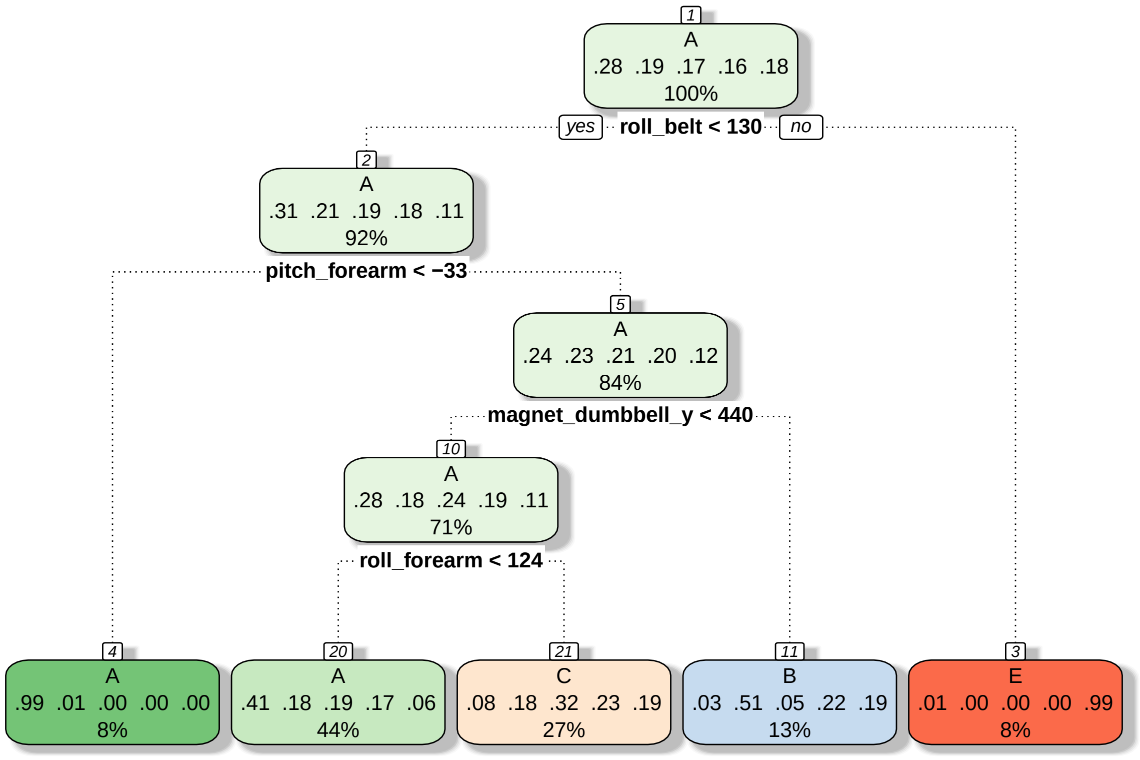
```
set.seed(213)
inTrain <- createDataPartition(keeptrain$classe, p = 0.5,list=FALSE)
training <- keeptrain[inTrain,]
testing <- keeptrain[-inTrain,]
```

We use a recursive parititioning machine learning algorithm to produce a model on the training data set. 5-fold cross-validation is used to keep the algortihm from overfitting. Recursive partitioning has the advantage of being relatively fast for a data set this large.

```
fitControl <- trainControl( method = "cv", number = 5)
rpartFit <- train(classe ~ ., data = training, method = "rpart", trControl = fitControl)
```

The model can be shown graphically as a decision tree.

```
## 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.
```



Rattle 2014−Oct−26 13:16:23 xan

On the plus side, our model is very simple with only 4 variables (out of 53) in use, which shows we have avoided overfitting the data. However, the model is likely too simple since it will never predict classification "D". More tuning is needed.

We can compute the in-sample error rate to be about 50% by comparing the predicted classification with the known classifications. Rpart provides probabilities for each classification and row. We use the high probability value as the prediction for that observation.

```r
missClass = function(values,prediction){sum(prediction != values)/length(values)}

predTrain <- predict(rpartFit$finalModel, newdata=training)
bestTrain <- max.col(predTrain)
bestTrainClasse <- mapvalues(bestTrain, from = c(1,2,3,5), to = c("A", "B", "C", "E"))
missClass(training$classe, bestTrainClasse)
```

```
## [1] 0.5075
```

Though the accuracy is low, we expect to do about as well with real data since the model is so simple. Normally we expect the out-of-sample error rate to be worse than the in-sample error, though the worsening should be minimal with such a simple model. Indeed the testing data set also shows an error rate of about 50%.

With more time, we would explore other models and tuning values.