Practical Machine Learning - Course Project

Alasdair Hazen

Executive Summary

Six subject participated in a study of exercising with dumbells. The experiment examines, not the quantity, but the manner in which they performed the exercise. There were five manners in which the subjects could do the exercise. (A) exactly to the specification, (B) throwing the elbows to the front, (C) lifting the dumbell only halfway, (D) lowering the dumbbell only halfway, and (E) throwing the hips to the front. The goal of this project is to analyse the data and predict which manner they performed the exercise based on the data. For this we will construct several models and select the best one according to accuracy.

Question

By analysing the data from accelerometers on the belt, forearm, arm, and dumbell using an algorithm, can the appropriate activity quality class be predicted?

Input Data

1. Set libraries
    - caret
    - rattle
    - rpart
    - randomForest
    - class
2. Download data
3. Read data into datasets
4. Drop unnecessary columns
    - NAs
    - blanks
    - errors
5. Split testing data into smaller group(s) to build a model and test the model

```r
# Check packages are installed
# any(grepl("caret", installed.packages()))
# any(grepl("rattle", installed.packages()))
# any(grepl("rpart", installed.packages()))
# any(grepl("rpart.plot", installed.packages()))
# any(grepl("randomForest", installed.packages()))

# Set libraries
library(caret)
library(rattle)
library(rpart)
library(rpart.plot)
library(randomForest)
library(class)
```

```r
# Download data
if (!file.exists("pml-training.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv",
                destfile = "pml-training.csv")
}

if (!file.exists("pml-testing.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv",
                destfile = "pml-testing.csv")
}
```

```r
# Read data into datasets
pmlTrainingData <- read.csv("pml-training.csv", na.strings=c("NA",""), header=T
RUE)
pmlTestingData <- read.csv("pml-testing.csv", na.strings=c("NA",""), header=TRU
E)
```

```r
# Drop unnecessary columns
pmlTrainingData <- pmlTrainingData[,!(names(pmlTrainingData) %in% drop)]
pmlTrainingData <- pmlTrainingData[,8:length(colnames(pmlTrainingData))]

pmlTestingData <- pmlTestingData[,!(names(pmlTestingData) %in% drop)]
pmlTestingData <- pmlTestingData[,8:length(colnames(pmlTestingData))]
```

We will do a quick check that our testing data and our training data columns are the same.

```r
# Show remaining columns.
colnames(pmlTrainingData)
```

```
##  [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"
```

```
colnames(pmlTestingData)
```

```
##  [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"     "problem_id"
```

```
# show classe
table(pmlTrainingData$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
# show probability
round(prop.table(table(pmlTrainingData$classe)) * 100, digits = 1)
```

```
##
##    A    B    C    D    E
## 28.4 19.4 17.4 16.4 18.4
```

Here we can see that the last column differes. In Training data the last column is classe, whereas in Testing data the column is problem_id. We will need to adjust for this.

There are several ways we can partition the data. We can use folds, partitions, etc. and a variety of variations and combinations. However, after 16 tries I have decided that the simplest 60/40 partition is just as effective as a more complicated derivative.

```
# Divide each of these 4 sets into training (60%) and test (40%) sets.
set.seed(1979)
inTrain <- createDataPartition(y=pmlTrainingData$classe, p=0.6, list=FALSE)
training <- pmlTrainingData[inTrain,]
testing <- pmlTrainingData[-inTrain,]
```

Features

1. Convert/Create covariates
    - Covariates that have no variability (NearZeroVar)
2. Check for overfitting

```
# Check NearZeroVar
nzv <- nearZeroVar(pmlTrainingData, saveMetrics=TRUE)
nzv
```

```
##                       freqRatio percentUnique zeroVar   nzv
## roll_belt              1.101904     6.7781062   FALSE FALSE
## pitch_belt             1.036082     9.3772296   FALSE FALSE
## yaw_belt               1.058480     9.9734991   FALSE FALSE
## total_accel_belt       1.063160     0.1477933   FALSE FALSE
## gyros_belt_x           1.058651     0.7134849   FALSE FALSE
## gyros_belt_y           1.144000     0.3516461   FALSE FALSE
## gyros_belt_z           1.066214     0.8612782   FALSE FALSE
## accel_belt_x           1.055412     0.8357966   FALSE FALSE
## accel_belt_y           1.113725     0.7287738   FALSE FALSE
## accel_belt_z           1.078767     1.5237998   FALSE FALSE
## magnet_belt_x          1.090141     1.6664968   FALSE FALSE
## magnet_belt_y          1.099688     1.5187035   FALSE FALSE
## magnet_belt_z          1.006369     2.3290184   FALSE FALSE
## roll_arm              52.338462    13.5256345   FALSE FALSE
## pitch_arm             87.256410    15.7323412   FALSE FALSE
## yaw_arm               33.029126    14.6570176   FALSE FALSE
## total_accel_arm        1.024526     0.3363572   FALSE FALSE
## gyros_arm_x            1.015504     3.2769341   FALSE FALSE
## gyros_arm_y            1.454369     1.9162165   FALSE FALSE
## gyros_arm_z            1.110687     1.2638875   FALSE FALSE
## accel_arm_x            1.017341     3.9598410   FALSE FALSE
## accel_arm_y            1.140187     2.7367241   FALSE FALSE
## accel_arm_z            1.128000     4.0362858   FALSE FALSE
## magnet_arm_x           1.000000     6.8239731   FALSE FALSE
## magnet_arm_y           1.056818     4.4439914   FALSE FALSE
## magnet_arm_z           1.036364     6.4468454   FALSE FALSE
## roll_dumbbell          1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell         2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell           1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell   1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x       1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y       1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z       1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x       1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y       1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z       1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x      1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y      1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z      1.020833     3.4451126   FALSE FALSE
## roll_forearm          11.589286    11.0895933   FALSE FALSE
## pitch_forearm         65.983051    14.8557741   FALSE FALSE
## yaw_forearm           15.322835    10.1467740   FALSE FALSE
## total_accel_forearm    1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x        1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y        1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z        1.122917     1.5645704   FALSE FALSE
## accel_forearm_x        1.126437     4.0464784   FALSE FALSE
```

```
## accel_forearm_y        1.059406     5.1116094   FALSE FALSE
## accel_forearm_z        1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x       1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y       1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z       1.000000     8.5771073   FALSE FALSE
## classe                 1.469581     0.0254816   FALSE FALSE
```

Near zero variance showing all false shows our data is clean after we removed superfluous columns. No more cleaning is required.

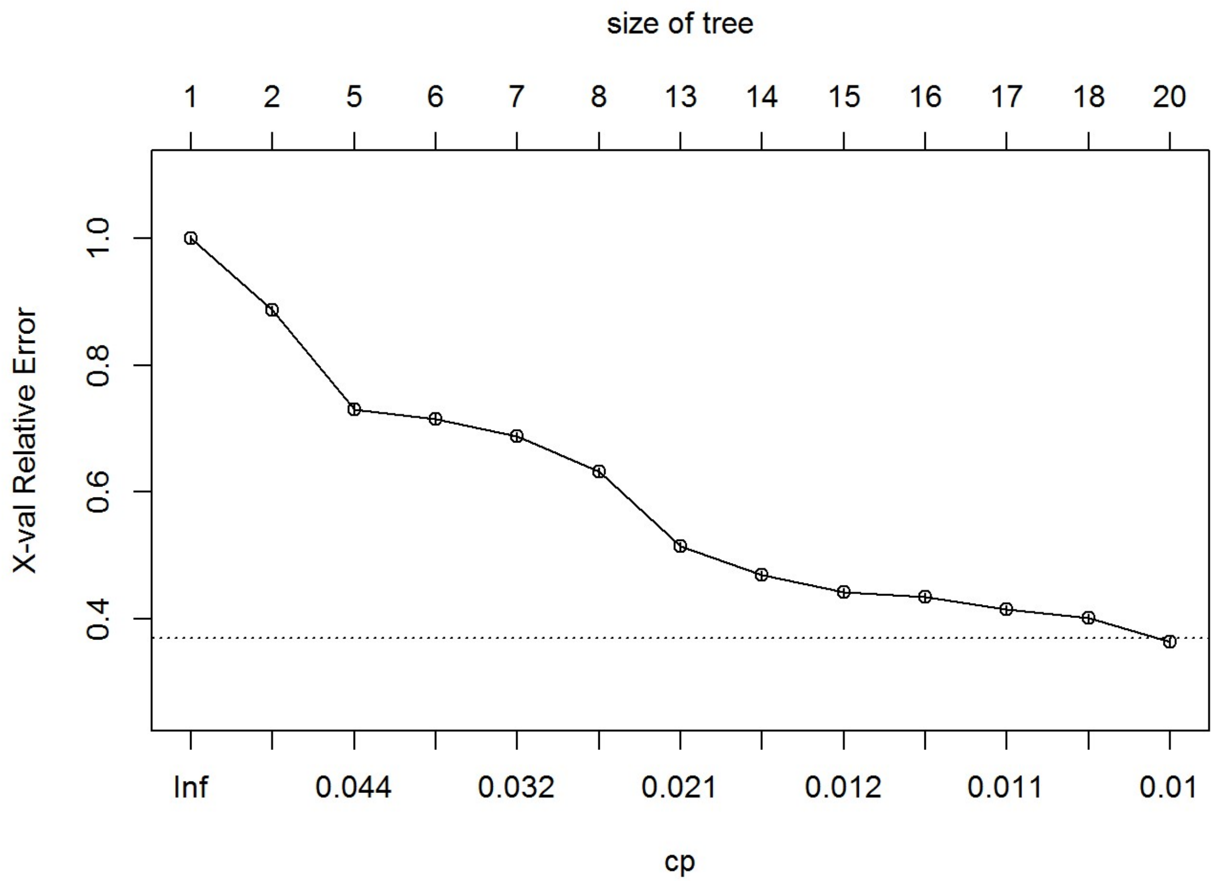
Algorithm

1. Cart Modeling via rpart
2. classification tree with (method=rpart)
3. Regression
4. Random forest (method=rf)

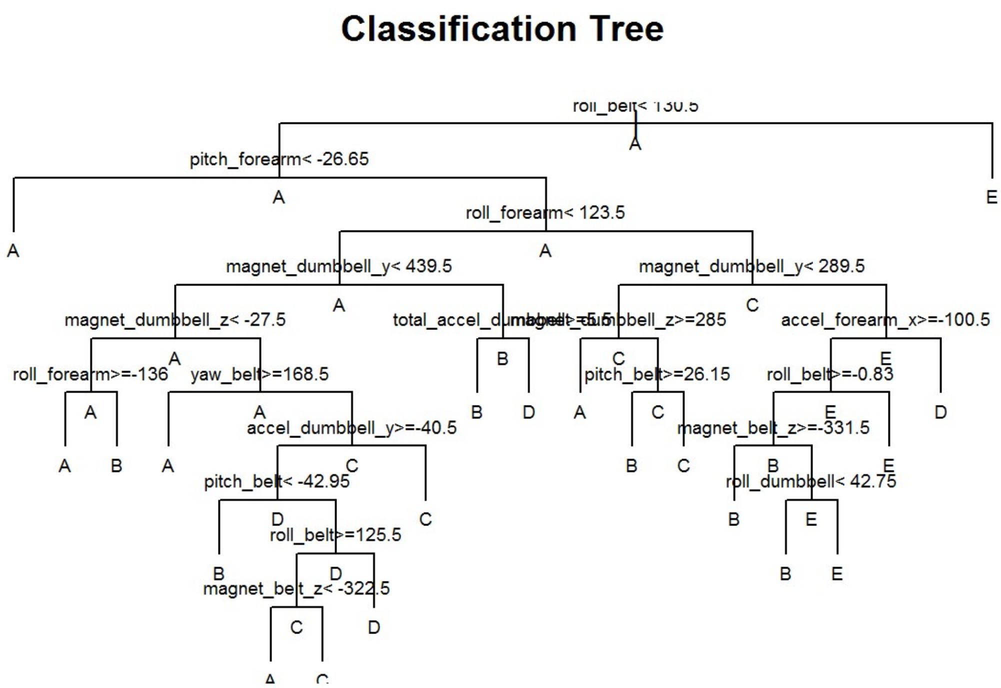
```
# grow tree
fit <- rpart(classe ~ .,method="class",data=training)
printcp(fit) # display the results
```

```
##
## Classification tree:
## rpart(formula = classe ~ ., data = training, method = "class")
##
## Variables actually used in tree construction:
##  [1] accel_dumbbell_y     accel_forearm_x      magnet_belt_z
##  [4] magnet_dumbbell_y    magnet_dumbbell_z    pitch_belt
##  [7] pitch_forearm        roll_belt            roll_dumbbell
## [10] roll_forearm         total_accel_dumbbell yaw_belt
##
## Root node error: 8428/11776 = 0.71569
##
## n= 11776
##
##          CP nsplit rel error  xerror      xstd
## 1  0.113075      0   1.00000 1.00000 0.0058081
## 2  0.050724      1   0.88692 0.88728 0.0061987
## 3  0.038087      4   0.72057 0.73113 0.0064309
## 4  0.034053      5   0.68249 0.71583 0.0064359
## 5  0.030612      6   0.64843 0.68747 0.0064371
## 6  0.022781      7   0.61782 0.63253 0.0064090
## 7  0.018747     12   0.49573 0.51448 0.0062102
## 8  0.012696     13   0.47698 0.46868 0.0060792
## 9  0.011984     14   0.46429 0.44198 0.0059878
## 10 0.011391     15   0.45230 0.43439 0.0059596
## 11 0.010916     16   0.44091 0.41445 0.0058813
## 12 0.010679     17   0.43000 0.40081 0.0058236
## 13 0.010000     19   0.40864 0.36308 0.0056467
```

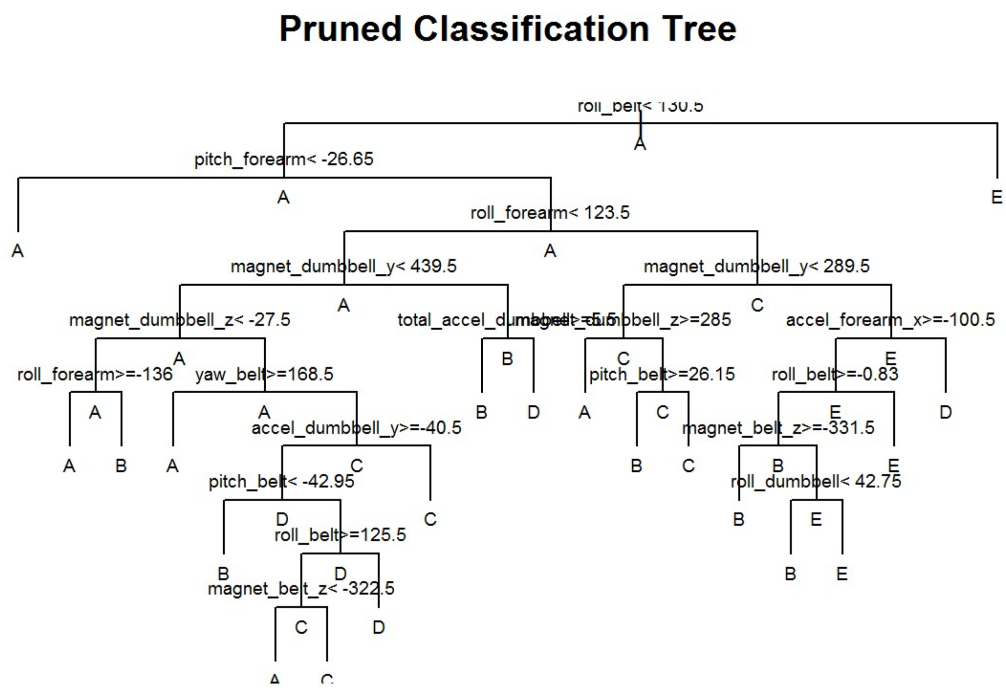
```
# plot tree
plotcp(fit) # visualize cross-validation results
```




```
# summary(fit) # detailed summary of splits   (For testing only. Lengthy outpu
t)
plot(fit, uniform=TRUE,
   main="Classification Tree")
text(fit, use.n=FALSE, all=TRUE, cex=.6)
```




```
# prune the tree
pfit<- prune(fit, cp= fit$testing[which.min(fit$testing[,"xerror"]),"CP"])
# plot the pruned tree
plot(pfit, uniform=TRUE,
   main="Pruned Classification Tree")
text(pfit, use.n=FALSE, all=TRUE, cex=.6)
```



Parameters

1. cross-validation
2. bootstrapping

```
# Random Forest prediction of training data
fit <- randomForest(classe ~ .,   data=training)
print(fit) # view results
```

```
## 
## Call:
##  randomForest(formula = classe ~ ., data = training) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
## 
##         OOB estimate of  error rate: 0.65%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3345    3    0    0    0 0.0008960573
## B   16 2258    5    0    0 0.0092145678
## C    0   16 2032    6    0 0.0107108082
## D    0    0   20 1909    1 0.0108808290
## E    0    0    0   10 2155 0.0046189376
```

```
importance(fit) # importance of each predictor
```

```
##                      MeanDecreaseGini
## roll_belt                   754.89965
## pitch_belt                  406.49451
## yaw_belt                    520.85462
## total_accel_belt            119.80911
## gyros_belt_x                 61.53216
## gyros_belt_y                 66.83782
## gyros_belt_z                180.41815
## accel_belt_x                 75.61335
## accel_belt_y                 74.75943
## accel_belt_z                249.44779
## magnet_belt_x               151.12180
## magnet_belt_y               230.18909
## magnet_belt_z               227.10018
## roll_arm                    182.20775
## pitch_arm                    99.67084
## yaw_arm                     147.51201
## total_accel_arm              59.51907
## gyros_arm_x                  77.71598
## gyros_arm_y                  81.91224
## gyros_arm_z                  38.71183
## accel_arm_x                 137.14882
## accel_arm_y                  93.25851
## accel_arm_z                  75.99641
## magnet_arm_x                147.90905
## magnet_arm_y                128.33402
## magnet_arm_z                115.48929
## roll_dumbbell               251.42958
## pitch_dumbbell              105.70308
## yaw_dumbbell                164.72312
## total_accel_dumbbell        148.47264
## gyros_dumbbell_x             80.35083
## gyros_dumbbell_y            150.19494
## gyros_dumbbell_z             52.67296
## accel_dumbbell_x            150.41538
## accel_dumbbell_y            242.86676
## accel_dumbbell_z            194.76933
## magnet_dumbbell_x           297.72593
## magnet_dumbbell_y           399.45126
## magnet_dumbbell_z           445.67412
## roll_forearm                383.99512
## pitch_forearm               475.21346
## yaw_forearm                 111.62573
## total_accel_forearm          67.13098
## gyros_forearm_x              48.07885
## gyros_forearm_y              79.73248
## gyros_forearm_z              50.33325
## accel_forearm_x             198.90877
```

```
## accel_forearm_y              94.57658
## accel_forearm_z             151.64496
## magnet_forearm_x            138.50751
## magnet_forearm_y            136.13522
## magnet_forearm_z            184.30248
```

Train Model

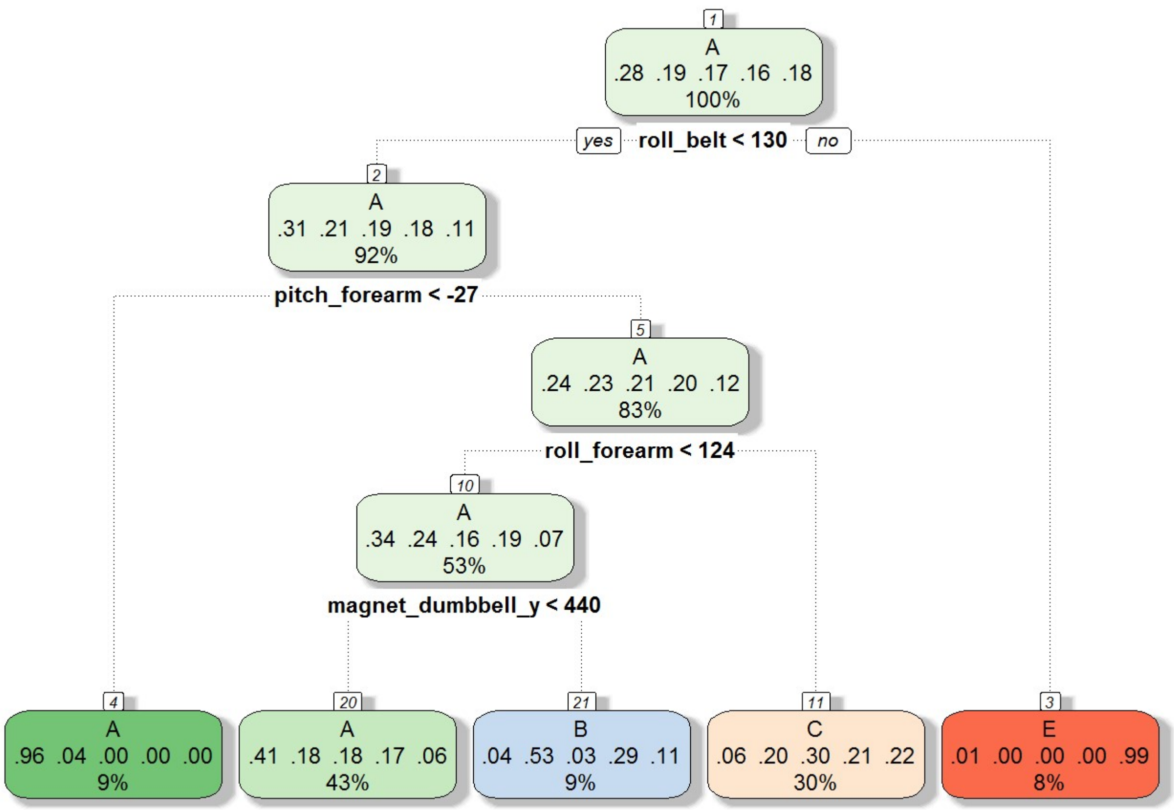
```
# Run rpart
set.seed(1979)
testModel <- train(training$classe ~ ., data = training, method="rpart")
print(testModel, digits=3)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0381  0.491     0.3334  0.0405       0.0658
##   0.0507  0.443     0.2558  0.0607       0.1028
##   0.1131  0.330     0.0711  0.0395       0.0593
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0381.
```

```
print(testModel$finalModel, digits=3)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8430 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130 10803 7460 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -26.6 1030   46 A (0.96 0.045 0 0 0) *
##      5) pitch_forearm>=-26.6 9773 7420 A (0.24 0.23 0.21 0.2 0.12)
##       10) roll_forearm< 124 6228 4100 A (0.34 0.24 0.16 0.19 0.069)
##         20) magnet_dumbbell_y< 440 5119 3030 A (0.41 0.18 0.18 0.17 0.061) *
##         21) magnet_dumbbell_y>=440 1109  518 B (0.04 0.53 0.029 0.29 0.11) *
##       11) roll_forearm>=124 3545 2460 C (0.063 0.2 0.3 0.21 0.22) *
##    3) roll_belt>=130 973   10 E (0.01 0 0 0 0.99) *
```

```
fancyRpartPlot(testModel$finalModel)
```



Rattle 2016-Jan-29 10:59:54 alasd

```
# Run with no extra features.
predictions <- predict(testModel, newdata=testing)
print(confusionMatrix(predictions, testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2032  679  645  591  213
##          B   36  359   21  218   91
##          C  160  480  702  477  470
##          D    0    0    0    0    0
##          E    4    0    0    0  668
##
## Overall Statistics
##
##                Accuracy : 0.4794
##                  95% CI : (0.4682, 0.4905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3191
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.23650  0.51316   0.0000  0.46325
## Specificity            0.6209  0.94216  0.75502   1.0000  0.99938
## Pos Pred Value         0.4885  0.49517  0.30668      NaN  0.99405
## Neg Pred Value         0.9457  0.83724  0.88015   0.8361  0.89211
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2590  0.04576  0.08947   0.0000  0.08514
## Detection Prevalence   0.5302  0.09240  0.29174   0.0000  0.08565
## Balanced Accuracy      0.7657  0.58933  0.63409   0.5000  0.73131
```

```
# Train with only preprocessing.
set.seed(1979)
testModel1 <- train(training$classe ~ .,  preProcess=c("center", "scale"), dat
a = training, method="rpart")
print(testModel1, digits=3)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0381  0.491     0.3334  0.0405       0.0658
##   0.0507  0.443     0.2558  0.0607       0.1028
##   0.1131  0.330     0.0711  0.0395       0.0593
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0381.
```

```
# Train with only cross validation.
set.seed(1979)
testModel2 <- train(training$classe ~ .,  trControl=trainControl(method = "c
v", number = 4), data = training, method="rpart")
print(testModel, digits=3)
```


```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0381  0.491     0.3334  0.0405       0.0658
##   0.0507  0.443     0.2558  0.0607       0.1028
##   0.1131  0.330     0.0711  0.0395       0.0593
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0381.
```

```
# Train with both preprocessing and cross validation.
set.seed(1979)
testModel2 <- train(training$classe ~ .,  preProcess=c("center", "scale"), trCo
ntrol=trainControl(method = "cv", number = 4), data = training, method="rpart")
print(testModel, digits=3)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0381  0.491     0.3334  0.0405       0.0658
##   0.0507  0.443     0.2558  0.0607       0.1028
##   0.1131  0.330     0.0711  0.0395       0.0593
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0381.
```

```
# Train with only cross validation.
set.seed(1979)
testModel3 <- train(training$classe ~ ., method="rf", trControl=trainControl(me
thod = "cv", number = 4), data=training)
print(testModel3, digits=3)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 8832, 8831, 8831, 8834
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.988     0.985  0.00210      0.00265
##   27    0.987     0.984  0.00154      0.00195
##   52    0.979     0.974  0.00358      0.00453
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Evaluation

Now that we've made our model, we put our subset test data into model.

```
# Run against testModel3.
predictions <- predict(testModel3, newdata=testing)
print(confusionMatrix(predictions, testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   17    0    0    0
##          B    0 1498   10    0    0
##          C    0    3 1355   21    1
##          D    0    0    3 1265    6
##          E    1    0    0    0 1435
##
## Overall Statistics
##
##                Accuracy : 0.9921
##                  95% CI : (0.9899, 0.9939)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.99
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9868   0.9905   0.9837   0.9951
## Specificity            0.9970   0.9984   0.9961   0.9986   0.9998
## Pos Pred Value         0.9924   0.9934   0.9819   0.9929   0.9993
## Neg Pred Value         0.9998   0.9968   0.9980   0.9968   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1909   0.1727   0.1612   0.1829
## Detection Prevalence   0.2865   0.1922   0.1759   0.1624   0.1830
## Balanced Accuracy      0.9983   0.9926   0.9933   0.9911   0.9975
```

```
# Run against 20 testing set provided by Professor Leek.
print(predict(testModel3, newdata=pmlTestingData))
```

```
##  [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
```

Conclusion

While a model based on Random Forest is slow and uses more processing time, the accuracy jumps from approximately 50% to over 95%, demonstrating that this is a better predictor.