Practical Machine Learning - Peer Graded Assignment

Summary

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.

The goal of you project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

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

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

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

```
inTrain <- createDataPartition(y=dataTraining$classe,p=0.6,list=FALSE)
myTrain <- dataTraining[inTrain, ]
myTest <- dataTraining[-inTrain, ]

dim(myTrain)
```

```
## [1] 11776   160
```

```
## Remove variables that are most NAs.
myTrainClean <- myTrain
for (i in 1:length(myTrain)) {
  if (sum(is.na(myTrain[ , i])) / nrow(myTrain) >= .75) {
    for (j in 1:length(myTrainClean)) {
      if (length(grep(names(myTrain[i]), names(myTrainClean)[j]))==1) {
        myTrainClean <- myTrainClean[ , -j]
      }
    }
  }
}

dim(myTrainClean)
```

```
## [1] 11776    60
```

```
## Remove unneccessary columns
myTrainingNew <- myTrainClean[,8:length(myTrainClean)]

## Remove near zero variables
nearZero <- nearZeroVar(myTrainingNew,saveMetrics=TRUE)
nearZero
```

```
##                     freqRatio percentUnique zeroVar   nzv
## roll_belt            1.030303    8.67866848   FALSE FALSE
## pitch_belt           1.042017   13.83322011   FALSE FALSE
## yaw_belt             1.029412   14.60597826   FALSE FALSE
## total_accel_belt     1.062986    0.23777174   FALSE FALSE
## gyros_belt_x         1.018980    1.05298913   FALSE FALSE
## gyros_belt_y         1.143265    0.55197011   FALSE FALSE
## gyros_belt_z         1.092219    1.34171196   FALSE FALSE
## accel_belt_x         1.084926    1.31623641   FALSE FALSE
## accel_belt_y         1.044229    1.13790761   FALSE FALSE
## accel_belt_z         1.164329    2.43716033   FALSE FALSE
```

```
## magnet_belt_x         1.058559    2.48811141   FALSE FALSE
## magnet_belt_y         1.154255    2.33525815   FALSE FALSE
## magnet_belt_z         1.033088    3.61752717   FALSE FALSE
## roll_arm             49.219512   19.69259511   FALSE FALSE
## pitch_arm            91.772727   22.51188859   FALSE FALSE
## yaw_arm              30.119403   21.34850543   FALSE FALSE
## total_accel_arm       1.005396    0.55197011   FALSE FALSE
## gyros_arm_x           1.033670    5.28192935   FALSE FALSE
## gyros_arm_y           1.290123    3.05706522   FALSE FALSE
## gyros_arm_z           1.215753    1.88519022   FALSE FALSE
## accel_arm_x           1.000000    6.36888587   FALSE FALSE
## accel_arm_y           1.133333    4.45822011   FALSE FALSE
## accel_arm_z           1.038462    6.43682065   FALSE FALSE
## magnet_arm_x          1.076923   11.21773098   FALSE FALSE
## magnet_arm_y          1.105263    7.18410326   FALSE FALSE
## magnet_arm_z          1.045455   10.55536685   FALSE FALSE
## roll_dumbbell         1.142857   87.63586957   FALSE FALSE
## pitch_dumbbell        2.740260   85.23267663   FALSE FALSE
## yaw_dumbbell          1.084507   87.11786685   FALSE FALSE
## total_accel_dumbbell  1.121655    0.34816576   FALSE FALSE
## gyros_dumbbell_x      1.096591    1.92764946   FALSE FALSE
## gyros_dumbbell_y      1.201729    2.24184783   FALSE FALSE
## gyros_dumbbell_z      1.163580    1.64741848   FALSE FALSE
## accel_dumbbell_x      1.028436    3.32031250   FALSE FALSE
## accel_dumbbell_y      1.211679    3.81283967   FALSE FALSE
## accel_dumbbell_z      1.076389    3.33729620   FALSE FALSE
## magnet_dumbbell_x     1.168421    8.85699728   FALSE FALSE
## magnet_dumbbell_y     1.252427    6.95482337   FALSE FALSE
## magnet_dumbbell_z     1.135135    5.55366848   FALSE FALSE
## roll_forearm         11.208738   15.17493207   FALSE FALSE
## pitch_forearm        69.969697   21.17017663   FALSE FALSE
## yaw_forearm          16.492857   14.09646739   FALSE FALSE
## total_accel_forearm   1.130081    0.56895380   FALSE FALSE
## gyros_forearm_x       1.135135    2.36073370   FALSE FALSE
## gyros_forearm_y       1.017937    5.96976902   FALSE FALSE
## gyros_forearm_z       1.171429    2.39470109   FALSE FALSE
## accel_forearm_x       1.185185    6.64911685   FALSE FALSE
## accel_forearm_y       1.046875    8.20312500   FALSE FALSE
## accel_forearm_z       1.076923    4.66202446   FALSE FALSE
## magnet_forearm_x      1.041667   12.08389946   FALSE FALSE
## magnet_forearm_y      1.207547   15.20889946   FALSE FALSE
## magnet_forearm_z      1.025641   13.31521739   FALSE FALSE
## classe                1.469065    0.04245924   FALSE FALSE
```

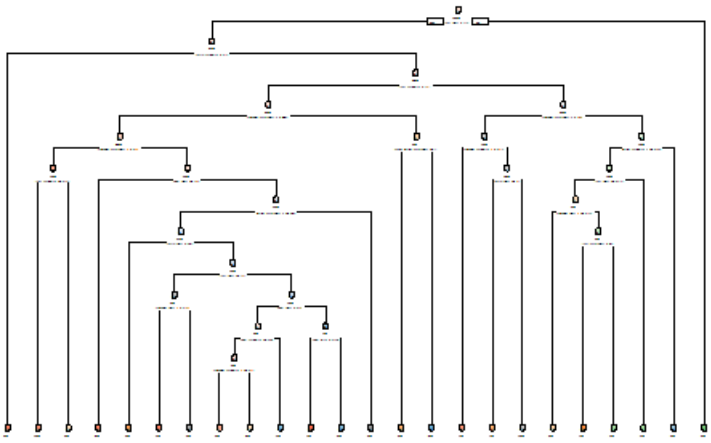
Random Decision Forest

```
## Random Forest
set.seed(123)
modelFit <- randomForest(classe~.,data=myTrainingNew)
print(modelFit)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = myTrainingNew)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3340    6    2    0    0 0.002389486
## B   11 2259    9    0    0 0.008775779
## C    0   14 2036    4    0 0.008763389
## D    0    0   22 1906    2 0.012435233
## E    0    0    3    7 2155 0.004618938
```

```
dataModel <- rpart(classe ~ .,data=myTrainingNew,method="class")
rpart.plot(dataModel,main="Figure 1: Classification",extra=100,under=TRUE,faclen=0)
```

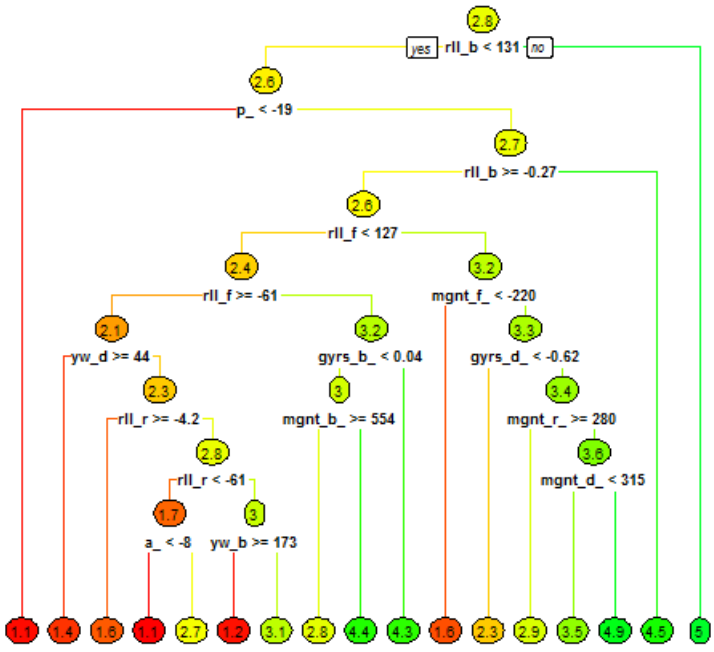
Figure 1: Classification



```
heat.tree <- function(tree, low.is.green = FALSE, ...)
  {
    # dots args passed to prp
    y <- tree$frame$yval
    if(low.is.green)
      y <- -y
    max <- max(y)
    min <- min(y)
    cols <- rainbow(99, end = .36)[
            ifelse(y > y[1], (y-y[1]) * (99-50) / (max-y[1]) + 50,
                   (y-min) * (50-1) / (y[1]-min) + 1)]
    prp(tree, branch.col = cols, box.col = cols, ...)
  }

treeModel <- rpart(classe ~ ., data=myTrainingNew,method="anova")
heat.tree(main="Figure 2: Heat Tree" ,treeModel,type=2,varlen=2,fallen.leaves=TRUE)
```

Figure 2: Heat Tree



```r
## Cross Validation
# Prediction 1
testPrediction <- predict(modelFit,myTest,type="class")
confusionMatrix(testPrediction,myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   13    0    0    0
##          B    0 1501    5    0    0
##          C    1    4 1363   23    0
##          D    0    0    0 1263    2
##          E    0    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9919, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9888   0.9963   0.9821   0.9986
## Specificity            0.9977   0.9992   0.9957   0.9997   1.0000
## Pos Pred Value         0.9942   0.9967   0.9799   0.9984   1.0000
## Neg Pred Value         0.9998   0.9973   0.9992   0.9965   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1913   0.1737   0.1610   0.1835
## Detection Prevalence   0.2860   0.1919   0.1773   0.1612   0.1835
## Balanced Accuracy      0.9986   0.9940   0.9960   0.9909   0.9993
```

```r
# Prediction 2
modelFit2 <- randomForest(classe ~. ,data=myTrainingNew,method="class")
testPrediction2 <- predict(dataModel,myTest,type="class")
confusionMatrix(testPrediction2,myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2010  216   21   59   32
##          B  108 1047  235  141  221
##          C   59  120 1034  192  182
##          D   28  124   78  855  140
##          E   27   11    0   39  867
##
## Overall Statistics
##
##                Accuracy : 0.7409
##                  95% CI : (0.731, 0.7506)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6716
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9005   0.6897   0.7558   0.6649   0.6012
## Specificity            0.9416   0.8886   0.9146   0.9436   0.9880
## Pos Pred Value         0.8597   0.5976   0.6515   0.6980   0.9184
## Neg Pred Value         0.9597   0.9227   0.9466   0.9349   0.9167
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2562   0.1334   0.1318   0.1090   0.1105
## Detection Prevalence   0.2980   0.2233   0.2023   0.1561   0.1203
## Balanced Accuracy      0.9211   0.7892   0.8352   0.8042   0.7946
```