Prediction_Assignment_Writeup

Executive summary

Outlined below I used personal activity data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. These participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. I took this data and constructed a machine learning algorithm that allowed me to predict accurately the manner in which they exercised. Ultimately the random forest approach proved to be the best being able accurately predict all 20 entries in my test data set.

More information on this study can be found here: http://groupware.les.inf.puc-rio.br/har

Load libraries and results Function

Session info

```
sessionInfo()
```

```
## R version 3.1.0 (2014-04-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.8
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.5.5 formatR_1.0    stringr_0.6.2  tools_3.1.0
```

```
library(caret)
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```
library(randomForest)
```

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

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(RCurl)
```

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

```
## Loading required package: bitops
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.1.2
```

```
library(rattle)
```

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

Setting a seed and loading in our data

```
set.seed(1234)

PML_training <- read.csv(file="C:\\coursera\\ML\\pml-training.csv", header=T)

PML_testing <- read.csv(file="C:\\coursera\\ML\\pml-testing.csv", header=T)
```

Data Preprocessing and Cleaning

```
#Now lets take a look at the data

str(PML_training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
## $ min_yaw_arm           : int  NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_roll_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_pitch_arm   : num  NA NA NA NA NA NA NA NA NA NA ...
## $ amplitude_yaw_arm     : int  NA NA NA NA NA NA NA NA NA NA ...
## $ roll_dumbbell         : num  13.1 13.1 12.9 13.4 13.4 ...
## $ pitch_dumbbell        : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
## $ yaw_dumbbell          : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
## $ kurtosis_roll_dumbbell : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ kurtosis_yaw_dumbbell  : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_roll_dumbbell : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ skewness_yaw_dumbbell  : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
## $ max_roll_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
## $ max_picth_dumbbell    : num  NA NA NA NA NA NA NA NA NA NA ...
## $ max_yaw_dumbbell      : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ min_roll_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
## $ min_pitch_dumbbell    : num  NA NA NA NA NA NA NA NA NA NA ...
## $ min_yaw_dumbbell      : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
## $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
# we see there are many NAs and "#DIV/0!" valeus generating a lot noise . Lets clean this up to create a tidy data set. Whatever we do to training we should also perform on testing

PML_training[PML_training==""] <- NA

PML_training[PML_training=='"#DIV/0!"'] <- NA

PML_training <- PML_training[,colSums(is.na(PML_training)) < .5 * nrow(PML_training)]


# lets remove columns we do not need

PML_train_clean <- PML_training[,c(-1:-7)]

dim(PML_train_clean)
```

```
## [1] 19622    53
```

Building a Model with Cross Validation

Now lets build a machine learning algorithm to predict activity quality from activity monitors Lets try two models: Trees and Random Forests. We will apply a cross validation when building our model.

Regression Trees

```
#### predict with trees

modFit <- train(classe ~ .,method="rpart",data=PML_train_clean,  trControl = trainControl(method = "cv"))
```

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

```
print(modFit$finalModel)
```

```
## n= 19622
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 19622 14042 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 17977 12411 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1578    10 A (0.99 0.0063 0 0 0) *
##      5) pitch_forearm>=-33.95 16399 12401 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 13870  9953 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 8643  5131 A (0.41 0.18 0.18 0.17 0.061) *
##         21) roll_forearm>=123.5 5227  3500 C (0.077 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 2529  1243 B (0.032 0.51 0.043 0.22 0.19) *
##    3) roll_belt>=130.5 1645    14 E (0.0085 0 0 0 0.99) *
```

```
print(modFit, digits =3)
```

```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 17661, 17659, 17660, 17658, 17660, 17660, ...
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0357  0.504     0.3527  0.0166       0.0222
```

```
##   0.0600  0.441     0.2502  0.0664      0.1114
##   0.1152  0.323     0.0583  0.0405      0.0616
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0357.
```

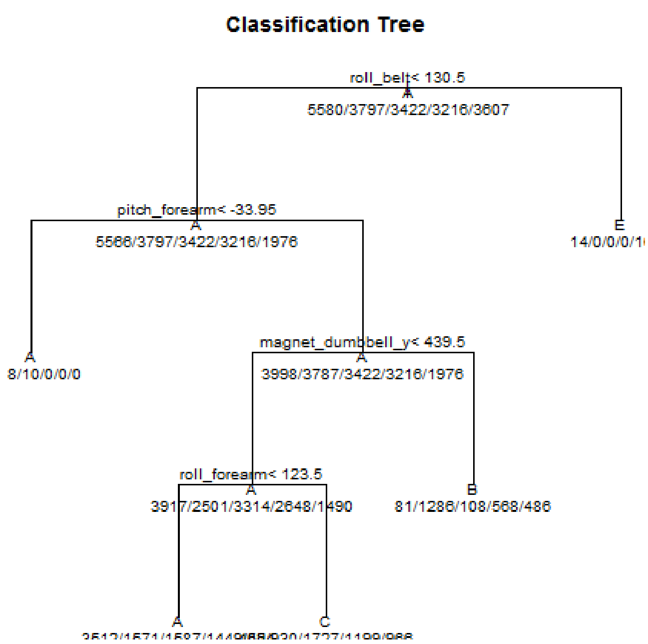
```
confusionMatrix(PML_train_clean$classe, predict(modFit, PML_train_clean))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5080   81  405    0   14
##          B 1581 1286  930    0    0
##          C 1587  108 1727    0    0
##          D 1449  568 1199    0    0
##          E  524  486  966    0 1631
##
## Overall Statistics
##
##                Accuracy : 0.4956
##                  95% CI : (0.4885, 0.5026)
##     No Information Rate : 0.5209
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3407
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4970  0.50850  0.33040       NA  0.99149
## Specificity            0.9468  0.85310  0.88225   0.8361  0.89008
## Pos Pred Value         0.9104  0.33869  0.50468       NA  0.45218
## Neg Pred Value         0.6339  0.92145  0.78395       NA  0.99913
## Prevalence             0.5209  0.12889  0.26638   0.0000  0.08383
## Detection Rate         0.2589  0.06554  0.08801   0.0000  0.08312
## Detection Prevalence   0.2844  0.19351  0.17440   0.1639  0.18382
## Balanced Accuracy      0.7219  0.68080  0.60633       NA  0.94079
```

```
#Accuracy of 49.56 % is not so good.

plot(modFit$finalModel, uniform=TRUE, main="Classification Tree")

text(modFit$finalModel, use.n=TRUE, all=TRUE, cex=.8)
```



Random Forests

```
# Since 49% was not so great lets try to predict with random forests now and add cross validation with trcontrol and cv as the method

model <- train(classe ~ .,method="rf",data=PML_train_clean,  trControl = trainControl(method = "cv"))

print(model$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5576    3    0    0    1 0.0007168459
## B   18 3774    5    0    0 0.0060574137
## C    0   13 3400    9    0 0.0064289889
## D    0    0   23 3190    3 0.0080845771
## E    0    1    4    5 3597 0.0027723870
```

```
print(model, digits =3)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 17661, 17661, 17659, 17660, 17659, 17660, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.995     0.994  0.00150      0.00189
##   27    0.995     0.994  0.00125      0.00158
##   52    0.990     0.988  0.00259      0.00327
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
confusionMatrix(PML_train_clean$classe, predict(model, PML_train_clean))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Results

Now that we concluded Random Forest is the right choice lets take our newly contrusted Machine Algorithm to predict activity quality from activity monitors with the test data. We can see from the confusion matrix above that our in sample accuracy with the training data is (0.995). We used 10 fold cross validation to predict the out of sample error.The final value used for the model was mtry of 2.

```
#now lets generate our results off our model and the test data set.
answers <- predict(model,newdata=PML_testing)


#lets take a look at our results
summary(answers)
```

```
## A B C D E
## 7 8 1 1 3
```

```
##Now lets write out our answers into individual files for the submission
```

```
#go to my answers directory
setwd('C:\\coursera\\ML\\answers')

pml_write_files(answers)
####
```