Practical Machine Learning

Introduction:

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The data for this project comes from this source: http://groupware.les.inf.puc-rio.br/har.

Loading in data

```
raw_training <- read.csv('./pml-training.csv', header=T)
raw_testing <- read.csv('./pml-testing.csv', header=T)
```

Loading the caret packages

```
library(caret)

## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.4
```

Previewing the data to see if any columns are empty or can be excluded

```
str(raw_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]
```

I need to remove the columns that are empty or have NAs. I will not be including them in the model.

```
cleanup <- sapply(names(raw_testing), function(x)
all(is.na(raw_testing[,x])==TRUE))
nonames <- names(cleanup)[cleanup==FALSE]
nonames <- nonames [-(1:7)]
nonames <- nonames [1:(length(nonames)-1)]
```

Now I will separate the training data into 2 sets.One set “70%” is used for the purpose of training and building the Random Forest model, and the other “30%” will be used to cross validate the model.

```
training_sample <- createDataPartition(y=raw_training$classe, p=0.7,
list=FALSE)
training <- raw_training[training_sample, ]
testing <- raw_training[-training_sample, ]
```

Cross Validating the model to use specific splits of data improve the accuracy.

```
fitControl <- trainControl(method='cv', number = 3)
```

Now I will build the model using the Random Forest model because I am the familiar with the model

```
rfmodel <- train(
                 classe ~ .,
                 data=training[, c('classe', nonames)],
                 trControl=fitControl,
                 method='rf',
                 ntree=200)
rfmodel

## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9871879  0.9837926
##   27    0.9874791  0.9841614
##   52    0.9795443  0.9741231
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

As you can see, the Random Forest Model has a high accuracy so I think this model will be sufficient to use for the prediction of the 20 testing variables.

Now I will use a Confusion Matrix to describe the model against the Testing set and the Training Set

```r
prediction <- predict(rfmodel, newdata=testing)
ConfusionMatrix <- confusionMatrix(prediction, testing$classe)
print(ConfusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    1 1131    7    0    1
##          C    1    5 1014   19    0
##          D    0    1    5  941    3
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9883   0.9761   0.9963
## Specificity            0.9995   0.9981   0.9949   0.9982   0.9992
## Pos Pred Value         0.9988   0.9921   0.9759   0.9905   0.9963
## Neg Pred Value         0.9995   0.9983   0.9975   0.9953   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1723   0.1599   0.1832
## Detection Prevalence   0.2845   0.1937   0.1766   0.1614   0.1839
## Balanced Accuracy      0.9992   0.9955   0.9916   0.9872   0.9977
```

```r
prediction <- predict(rfmodel, newdata=training)
ConfusionMatrix <- confusionMatrix(prediction, training$classe)
print(ConfusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
```

```
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

I want to build a decision tree model, even though the Random Forest model was strong.

```
model_tree <- train(
   classe ~ .,
   data=training[, c('classe', nonames)],
   trControl=fitControl,
   method='rpart')

 model_tree
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03316041  0.5153236  0.36616524
##   0.06069237  0.4513358  0.26715341
##   0.11585800  0.3385747  0.08291154
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03316041.
```

```r
prediction <- predict(model_tree, newdata=training)
ConfusionMatrix <- confusionMatrix(prediction, training$classe)
print(ConfusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3561 1089 1111 1023  375
##          B   68  917   79  404  328
##          C  265  652 1206  825  671
##          D    0    0    0    0    0
##          E   12    0    0    0 1151
##
## Overall Statistics
##
##                Accuracy : 0.4976
##                  95% CI : (0.4892, 0.506)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3432
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9117  0.34500  0.50334   0.0000  0.45584
## Specificity            0.6340  0.92066  0.78723   1.0000  0.99893
## Pos Pred Value         0.4974  0.51058  0.33324      NaN  0.98968
## Neg Pred Value         0.9476  0.85420  0.88239   0.8361  0.89073
## Prevalence             0.2843  0.19349  0.17442   0.1639  0.18381
## Detection Rate         0.2592  0.06675  0.08779   0.0000  0.08379
## Detection Prevalence   0.5211  0.13074  0.26345   0.0000  0.08466
## Balanced Accuracy      0.7728  0.63283  0.64529   0.5000  0.72739
```

This model only has a 0.49 accuracy so I will use the RF Model to predict the 20 observations.

Finally, I will predict the classe of the 20 test observations in the raw testing data by using the model created.

```r
predictionTesting <- predict(rfmodel, newdata=raw_testing)
print(predictionTesting)
```

```
##  [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

I will use the results above to answer to 20 multiple choice questions. Based on the accuracy rates of the RF model, I think my results are fairly accurate to predict the sample observations. After taking themultiple choice quiz, I answered all of the questions correctly.