Practical Machine Learning Prediction Assignment Course Project

Prabhu Thaipulley - 2nd April 2017

Contents


```
## Run time: 2017-04-02 15:29:39
## R version: R version 3.3.2 (2016-10-31)
```

This document establishes a stepwise description of the analysis performed for the prediction assignment of the Coursera's Practical Machine Learning course. This project uses data from the accelerometers of fitness devices of six participants to determine the manner in which they performed a particular exercise.

Project Background

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).

Preparing the datasets

This document establishes a stepwise description of the analysis performed for the prediction assignment of the Coursera's Practical Machine Learning course. This project uses data from the accelerometers of fitness devices of six participants to determine the manner in which they performed a particular exercise.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

```
library(data.table)

url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
TrainingData <- fread(url)

# record the download date, as mentioned in lectures
DownloadDate <- date()
sink("/Users/tsprabhu/github/PML_CourseProject/PML_CourseProject_files/data/download_date_training.txt")
cat("Date training data downloaded: ")
```

```
## Date training data downloaded:
```

```
cat(DownloadDate)
```

```
## Sun Apr  2 15:29:42 2017
```

Load the testing data into a data table.

```
sink()

url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
TestData <- fread(url)

# record the download date, as mentioned in lectures
DownloadDate <- date()
sink("/Users/tsprabhu/github/PML_CourseProject/PML_CourseProject_files/data/download_date_testing.txt")
cat("Date testing data downloaded: ")
```

```
## Date testing data downloaded:
```

```
cat(DownloadDate)
```

```
## Sun Apr  2 15:29:43 2017
```

```
sink()
```

Which variables in the test dataset have zero NAs? Use this tip: finding columns with all missing values in r.

Identify predictor candidates in the testing dataset We need to identify variables in the test dataset without missing or NA values; these will be suitable predictor candidates.

```
isAnyMissing <- sapply(TestData, function (x) any(is.na(x) | x == ""))
isPredictor <- !isAnyMissing & grepl("belt|[^(fore)]arm|dumbbell|forearm", names(isAnyMissing))
predCandidates <- names(isAnyMissing)[isPredictor]
predCandidates
```

```
##  [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"
```

Subset the primary dataset to include only the **predictor candidates** and the outcome variable, `classe`.

```
varToInclude <- c("classe", predCandidates)
TrainingData <- TrainingData[, varToInclude, with=FALSE]
dim(TrainingData)
```

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

```
names(TrainingData)
```

```
##  [1] "classe"               "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"
```

Make `classe` into a factor.

```
TrainingData <- TrainingData[, classe := factor(TrainingData[, classe])]
TrainingData[, .N, classe]
```

```
##    classe    N
## 1:      A 5580
## 2:      B 3797
## 3:      C 3422
## 4:      D 3216
## 5:      E 3607
```

Split the dataset into a 60% training and 40% probing dataset.

```
library(caret)
```

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

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

```
seed <- as.numeric(as.Date("2017-04-03"))
set.seed(seed)
inTrain <- createDataPartition(TrainingData$classe, p=0.6)
DTrain <- TrainingData[inTrain[[1]]]
DProbe <- TrainingData[-inTrain[[1]]]
```

Preprocess the prediction variables by centering and scaling.

```
X <- DTrain[, predCandidates, with=FALSE]
preProc <- preProcess(X)
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - scaled (52)
```

```
XCS <- predict(preProc, X)
DTrainCS <- data.table(data.frame(classe = DTrain[, classe], XCS))
```

Apply the centering and scaling to the probing dataset.

```
X <- DProbe[, predCandidates, with=FALSE]
XCS <- predict(preProc, X)
DProbeCS <- data.table(data.frame(classe = DProbe[, classe], XCS))
```

Check for near zero variance.

```
nzv <- nearZeroVar(DTrainCS, saveMetrics=TRUE)
if (any(nzv$nzv)) nzv else message("No variables with near zero variance")
```

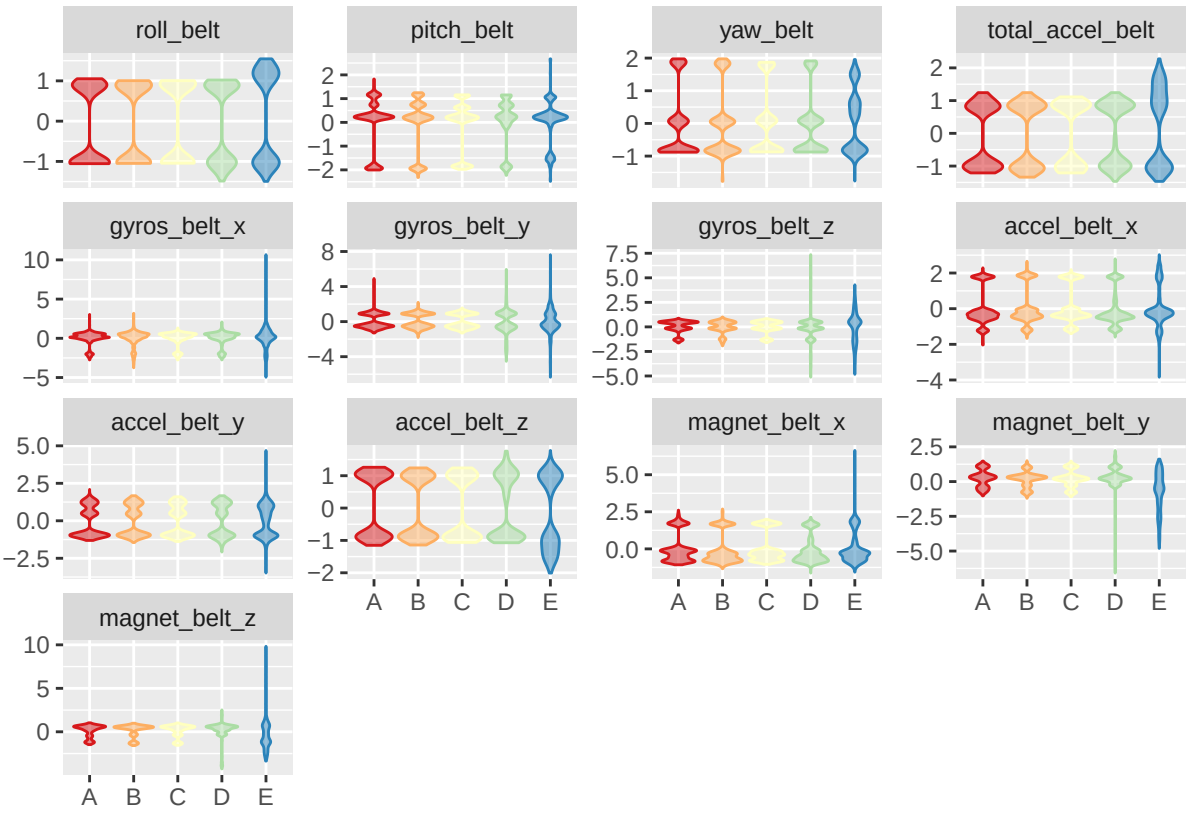
```
## No variables with near zero variance
```

Examine groups of prediction variables.

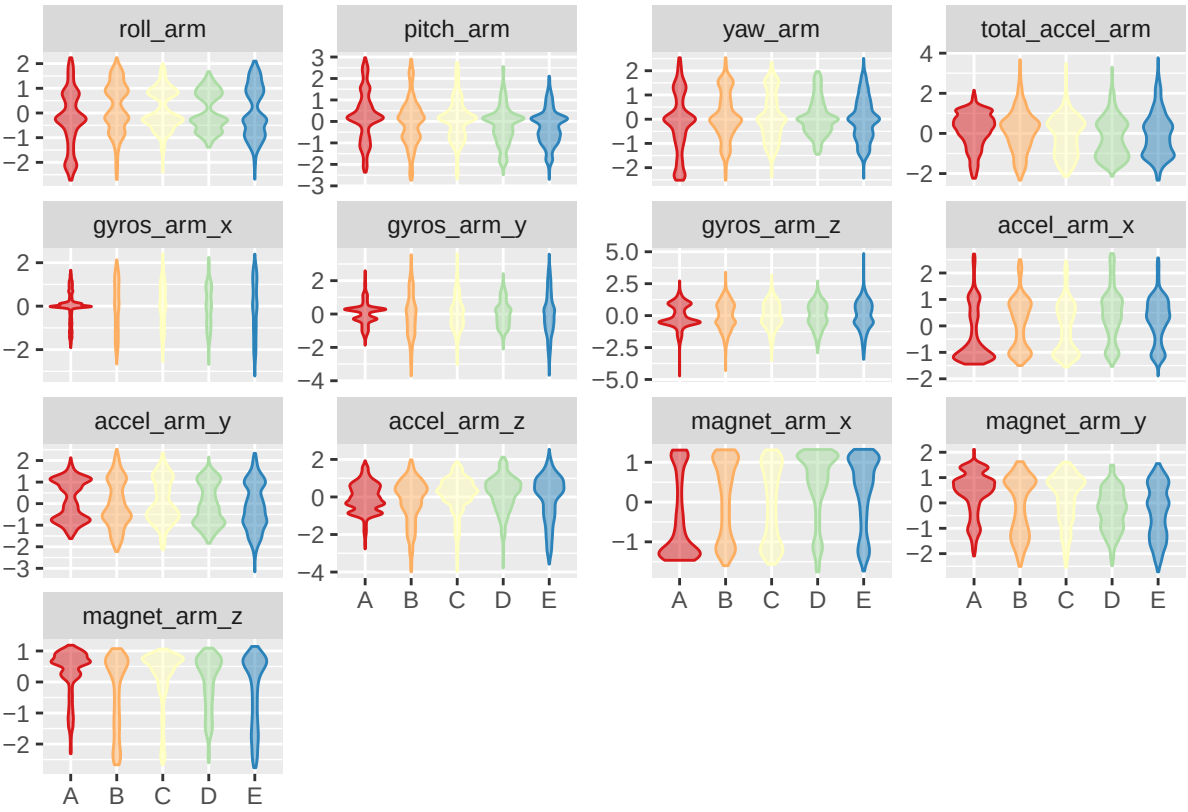
```
histGroup <- function (data, regex) {
  col <- grep(regex, names(data))
  col <- c(col, which(names(data) == "classe"))
  library(reshape2)
  n <- nrow(data)
  DMelted <- melt(data[, col, with=FALSE][, rownum := seq(1, n)], id.vars=c("rownum", "classe"))
  library(ggplot2)
  ggplot(DMelted, aes(x=classe, y=value)) +
    geom_violin(aes(color=classe, fill=classe), alpha=1/2) +
#     geom_jitter(aes(color=classe, fill=classe), alpha=1/10) +
#     geom_smooth(aes(group=1), method="gam", color="black", alpha=1/2, size=2) +
    facet_wrap(~ variable, scale="free_y") +
    scale_color_brewer(palette="Spectral") +
    scale_fill_brewer(palette="Spectral") +
    labs(x="", y="") +
    theme(legend.position="none")
}
histGroup(DTrainCS, "belt")
```

```
##
## Attaching package: 'reshape2'
```

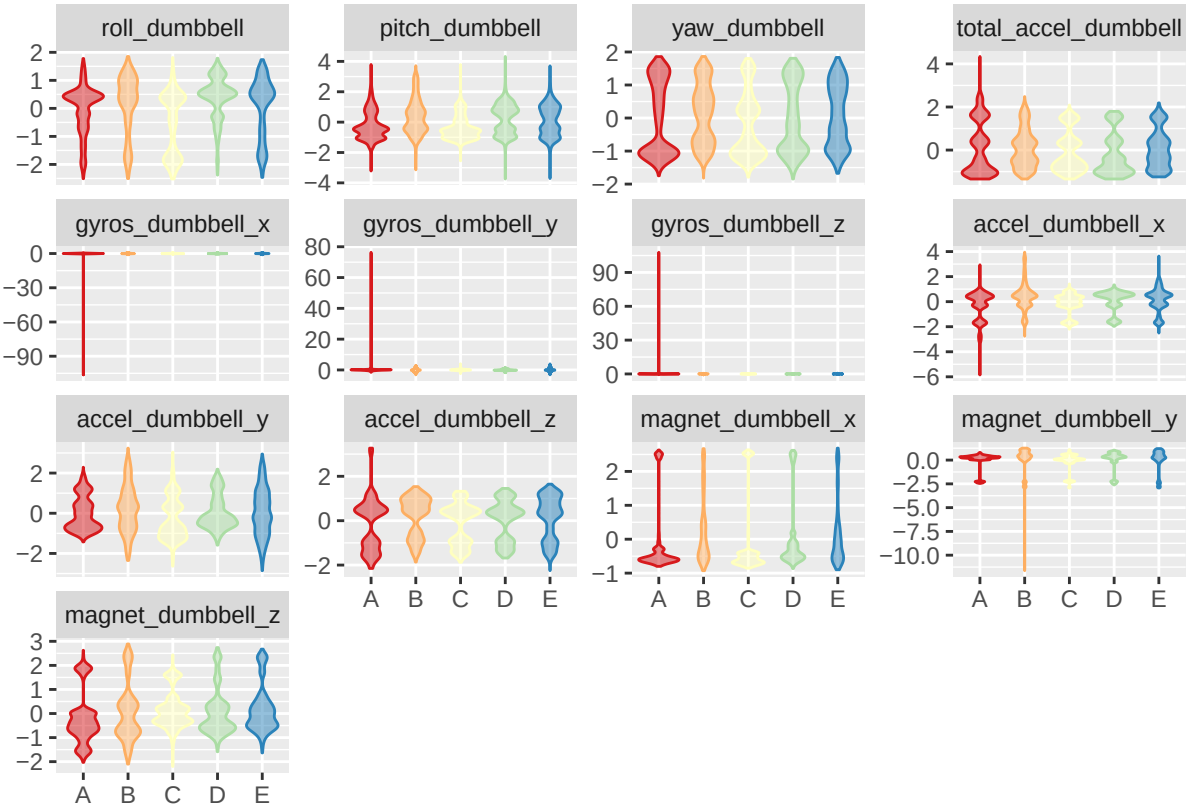
```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```



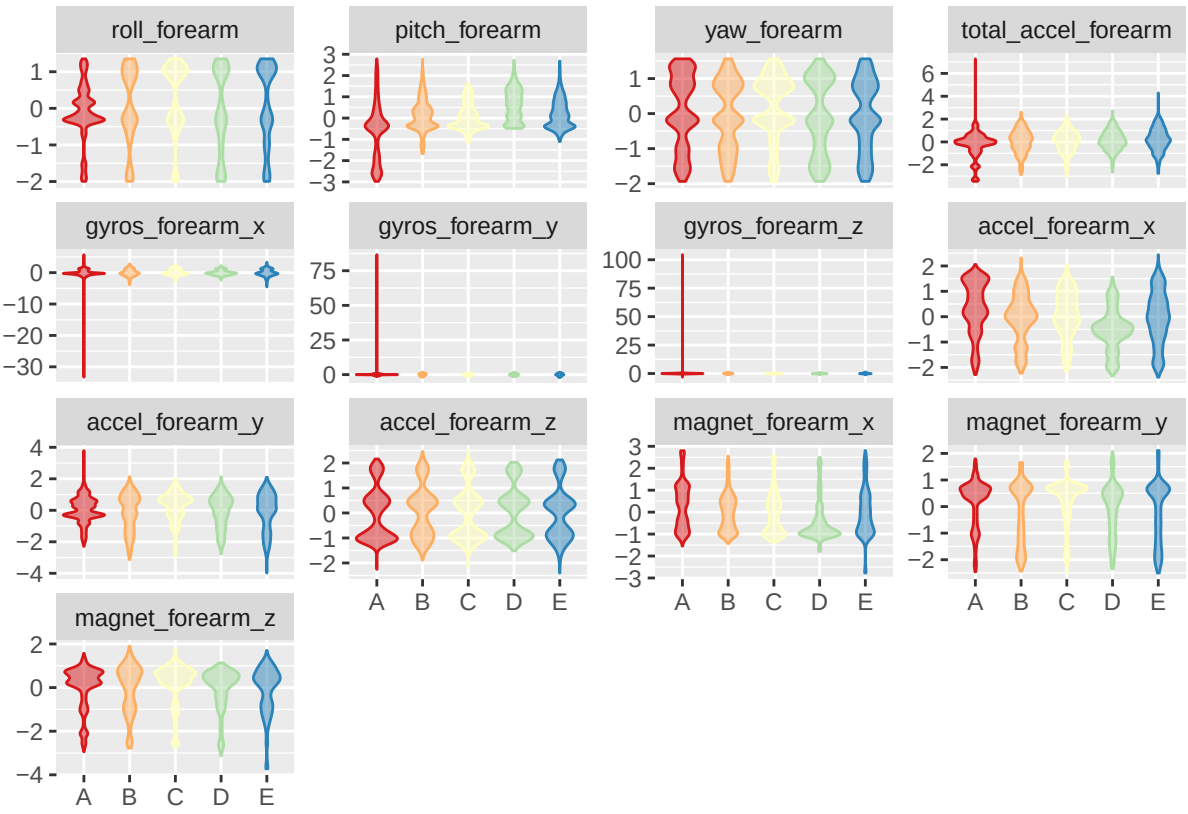

```
histGroup(DTrainCS, "[^(fore)]arm")
```



```
histGroup(DTrainCS, "dumbbell")
```




```
histGroup(DTrainCS, "forearm")
```



Train a prediction model

Using random forest, the out of sample error should be small. The error will be estimated using the 40% probing sample. I would be quite happy with an error estimate of 3% or less.

Set up the parallel clusters.

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
```

Set the control parameters.

```
ctrl <- trainControl(classProbs=TRUE,
                     savePredictions=TRUE,
                     allowParallel=TRUE)
```

Fit out model over the tuning parameters.

```
method <- "rf"
system.time(trainingModel <- train(classe ~ ., data=DTrainCS, method=method))
```

```
## Loading required package: randomForest

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

##     user   system  elapsed
##   36.083    0.247 1115.204
```

Finally, we terminate the clustering.

```
stopCluster(cl)
```

Evaluate the model on the training dataset

```
trainingModel
```

```
## Random Forest
##
## 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:
##
##   mtry  Accuracy   Kappa
##    2    0.9851924  0.9812643
##   27    0.9858432  0.9820886
##   52    0.9778534  0.9719799
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
final_result <- predict(trainingModel, DTrainCS)
confusionMatrix(final_result, DTrain[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## 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
```

Evaluate the model on the probing dataset

```
final_result <- predict(trainingModel, DProbeCS)
confusionMatrix(final_result, DProbeCS[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    8    0    0    0
##          B    2 1504   14    0    0
##          C    1    5 1349   15    0
##          D    0    1    5 1270    7
##          E    0    0    0    1 1435
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9903, 0.9943)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9908   0.9861   0.9876   0.9951
## Specificity            0.9986   0.9975   0.9968   0.9980   0.9998
## Pos Pred Value         0.9964   0.9895   0.9847   0.9899   0.9993
## Neg Pred Value         0.9995   0.9978   0.9971   0.9976   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1917   0.1719   0.1619   0.1829
## Detection Prevalence   0.2851   0.1937   0.1746   0.1635   0.1830
## Balanced Accuracy      0.9986   0.9941   0.9914   0.9928   0.9975
```

Display the final model

```
varImp(trainingModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          58.99
## yaw_belt               54.11
## pitch_belt             44.83
## magnet_dumbbell_y      44.11
## magnet_dumbbell_z      42.98
## roll_forearm           41.16
## accel_dumbbell_y       22.67
## roll_dumbbell          17.50
## accel_forearm_x        17.18
## magnet_belt_z          16.11
## magnet_dumbbell_x      15.83
## accel_belt_z           15.06
## magnet_forearm_z       14.22
## accel_dumbbell_z       14.18
## total_accel_dumbbell   13.03
## gyros_belt_z           11.02
## yaw_arm                10.84
## magnet_belt_y          10.69
## magnet_belt_x          10.54
```

```
trainingModel$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: 0.87%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B   20 2253    6    0    0 0.0114085125
## C    0   19 2028    7    0 0.0126582278
## D    0    2   31 1894    3 0.0186528497
## E    0    1    4    8 2152 0.0060046189
```

The estimated error rate is less than 1%.

Save training model object for later.

```
save(trainingModel, file="trainingModel.RData")
```

Predict on the test data

Load the training model.

```
load(file="trainingModel.RData", verbose=TRUE)
```

```
## Loading objects:
##   trainingModel
```

Get predictions and evaluate.

```
TestDataCS <- predict(preProc, TestData[, predCandidates, with=FALSE])
final_result <- predict(trainingModel, TestDataCS)
TestData <- cbind(final_result , TestData)
subset(TestData, select=names(TestData)[grep("belt|[^(fore)]arm|dumbbell|forearm", names(TestData), inv
```

```
##     final_result V1 user_name raw_timestamp_part_1 raw_timestamp_part_2
##  1:            B  1     pedro           1323095002               868349
##  2:            A  2    jeremy           1322673067               778725
##  3:            B  3    jeremy           1322673075               342967
##  4:            A  4    adelmo           1322832789               560311
##  5:            A  5    eurico           1322489635               814776
##  6:            E  6    jeremy           1322673149               510661
##  7:            D  7    jeremy           1322673128               766645
##  8:            B  8    jeremy           1322673076                54671
##  9:            A  9  carlitos           1323084240               916313
## 10:            A 10   charles           1322837822               384285
## 11:            B 11  carlitos           1323084277                36553
## 12:            C 12    jeremy           1322673101               442731
## 13:            B 13    eurico           1322489661               298656
## 14:            A 14    jeremy           1322673043               178652
```

```
## 15:            E 15    jeremy           1322673156               550750
## 16:            E 16    eurico           1322489713               706637
## 17:            A 17     pedro           1323094971               920315
## 18:            B 18  carlitos           1323084285               176314
## 19:            B 19     pedro           1323094999               828379
## 20:            B 20    eurico           1322489658               106658
##       cvtd_timestamp new_window num_window problem_id
##  1: 05/12/2011 14:23         no         74          1
##  2: 30/11/2011 17:11         no        431          2
##  3: 30/11/2011 17:11         no        439          3
##  4: 02/12/2011 13:33         no        194          4
##  5: 28/11/2011 14:13         no        235          5
##  6: 30/11/2011 17:12         no        504          6
##  7: 30/11/2011 17:12         no        485          7
##  8: 30/11/2011 17:11         no        440          8
##  9: 05/12/2011 11:24         no        323          9
## 10: 02/12/2011 14:57         no        664         10
## 11: 05/12/2011 11:24         no        859         11
## 12: 30/11/2011 17:11         no        461         12
## 13: 28/11/2011 14:14         no        257         13
## 14: 30/11/2011 17:10         no        408         14
## 15: 30/11/2011 17:12         no        779         15
## 16: 28/11/2011 14:15         no        302         16
## 17: 05/12/2011 14:22         no         48         17
## 18: 05/12/2011 11:24         no        361         18
## 19: 05/12/2011 14:23         no         72         19
## 20: 28/11/2011 14:14         no        255         20
```

Submission to Coursera

Write submission files to `Prediction Answers`.

```
pml_write_files = function(x){
  n = length(x)
  path <- "/Users/tsprabhu/github/PML_CourseProject/PML_CourseProject_files"
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=file.path(path, filename),quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(final_result)
```