Practical Machine Learning Proj

Alex L.

June 9, 2018

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are available here:

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

Choosing the prediction algorithm

Steps Taken

1.Tidy data. Remove columns with little/no data.

2.Create Training and test data from traing data for cross validation checking

3.Trial 3 methods Random Forrest, Gradient boosted model and Linear discriminant analysis

Fine tune model through combinations of above methods, reduction of input variables or similar. The fine tuning will take into account accuracy first and speed of analysis second.

```
library(caret)
```

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

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

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

```
library(ggplot2)
library(randomForest)
```

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

```
## 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
```

```
#read in training and testing data
train <- read.csv("C:/Users/viyeh/Documents/Resume/Coursera/Practical Machine Learning/pml-training.csv", na.strings=c("NA",
"#DIV/0!",""))
test <- read.csv("C:/Users/viyeh/Documents/Resume/Coursera/Practical Machine Learning/pml-testing.csv", na.strings=c("NA","#
DIV/0!",""))

names(train)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

```
str(train)
```

```
## '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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 -0.03 -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       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ 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        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
summary(train$classe)
```

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

```
#this is the outcome we want to predict
```

Split training/testing data

efore we do anything, we will set aside a subset of our training data for cross validation (40%).

```
#we want to predict the 'classe' variable using any other variable to predict with

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

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

```
dim(myTest)
```

```
## [1] 7846  160
```

Feature Selection

Now we can tranform the data to only include the variables we will need to build our model. We will remove variables with near zero variance, variables with mostly missing data, and variables that are obviously not useful as predictors.

```
#first we will remove variables with mostly NAs (use threshold of >75%)
mytrain_SUB <- myTrain
for (i in 1:length(myTrain)) {
  if (sum(is.na(myTrain[ , i])) / nrow(myTrain) >= .75) {
    for (j in 1:length(mytrain_SUB)) {
      if (length(grep(names(myTrain[i]), names(mytrain_SUB)[j]))==1) {
        mytrain_SUB <- mytrain_SUB[ , -j]
      }
    }
  }
}

dim(mytrain_SUB)
```

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

```
#names(mytrain_SUB)

#remove columns that are not predictors
mytrain_SUB2 <- mytrain_SUB[,8:length(mytrain_SUB)]

#remove variables with near zero variance
NZV <- nearZeroVar(mytrain_SUB2, saveMetrics = TRUE)
NZV #all false, none to remove
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.056391    8.73811141   FALSE FALSE
## pitch_belt            1.066038   13.68036685   FALSE FALSE
## yaw_belt              1.076923   14.55502717   FALSE FALSE
## total_accel_belt      1.079404    0.23777174   FALSE FALSE
## gyros_belt_x          1.086634    1.05298913   FALSE FALSE
## gyros_belt_y          1.176398    0.56046196   FALSE FALSE
## gyros_belt_z          1.061355    1.35869565   FALSE FALSE
## accel_belt_x          1.090717    1.31623641   FALSE FALSE
## accel_belt_y          1.152993    1.13790761   FALSE FALSE
## accel_belt_z          1.126482    2.41168478   FALSE FALSE
## magnet_belt_x         1.093458    2.49660326   FALSE FALSE
## magnet_belt_y         1.088083    2.39470109   FALSE FALSE
## magnet_belt_z         1.037037    3.60054348   FALSE FALSE
## roll_arm             49.476190   19.41236413   FALSE FALSE
## pitch_arm            83.160000   22.23165761   FALSE FALSE
## yaw_arm              28.465753   21.50135870   FALSE FALSE
## total_accel_arm       1.000000    0.56046196   FALSE FALSE
## gyros_arm_x           1.085809    5.26494565   FALSE FALSE
## gyros_arm_y           1.442675    3.04008152   FALSE FALSE
## gyros_arm_z           1.106918    1.94463315   FALSE FALSE
## accel_arm_x           1.092593    6.42832880   FALSE FALSE
## accel_arm_y           1.006944    4.40726902   FALSE FALSE
## accel_arm_z           1.178082    6.39436141   FALSE FALSE
## magnet_arm_x          1.036364   11.12432065   FALSE FALSE
## magnet_arm_y          1.035088    7.26052989   FALSE FALSE
## magnet_arm_z          1.092308   10.52989130   FALSE FALSE
## roll_dumbbell         1.012195   87.31317935   FALSE FALSE
## pitch_dumbbell        2.096386   85.12228261   FALSE FALSE
## yaw_dumbbell          1.092105   86.86311141   FALSE FALSE
## total_accel_dumbbell  1.064165    0.33967391   FALSE FALSE
## gyros_dumbbell_x      1.010811    1.97860054   FALSE FALSE
## gyros_dumbbell_y      1.303468    2.22486413   FALSE FALSE
## gyros_dumbbell_z      1.032345    1.61345109   FALSE FALSE
## accel_dumbbell_x      1.014851    3.43070652   FALSE FALSE
## accel_dumbbell_y      1.181818    3.84680707   FALSE FALSE
## accel_dumbbell_z      1.101449    3.33729620   FALSE FALSE
## magnet_dumbbell_x     1.055556    8.90794837   FALSE FALSE
## magnet_dumbbell_y     1.209524    6.91236413   FALSE FALSE
## magnet_dumbbell_z     1.041322    5.60461957   FALSE FALSE
## roll_forearm         11.311881   14.72486413   FALSE FALSE
## pitch_forearm        60.105263   21.10224185   FALSE FALSE
## yaw_forearm          14.358491   14.00305707   FALSE FALSE
## total_accel_forearm   1.118799    0.56046196   FALSE FALSE
## gyros_forearm_x       1.127869    2.32676630   FALSE FALSE
## gyros_forearm_y       1.061674    6.00373641   FALSE FALSE
## gyros_forearm_z       1.080268    2.36073370   FALSE FALSE
## accel_forearm_x       1.127273    6.64062500   FALSE FALSE
## accel_forearm_y       1.166667    8.27105978   FALSE FALSE
## accel_forearm_z       1.053191    4.65353261   FALSE FALSE
## magnet_forearm_x      1.192308   12.06691576   FALSE FALSE
## magnet_forearm_y      1.403846   15.24286685   FALSE FALSE
## magnet_forearm_z      1.026316   13.42561141   FALSE FALSE
## classe                1.469065    0.04245924   FALSE FALSE
```

```
keep <- names(mytrain_SUB2)
```

Random Forest Model

I chose to use the random forest model to build my machine learning algorithm as it is appropriate for a classification problem as we have. Based on class lectures this model tends to be more accurate than some other classification models.

Below I fit my model on my training data and then use my model to predict classe on my subset of data used for cross validation.

```
#fit model- RANDOM FOREST
set.seed(223)

modFit <- randomForest(classe~., data = mytrain_SUB2)
print(modFit)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = mytrain_SUB2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B   14 2255   10    0    0 0.0105309346
## C    0   19 2032    3    0 0.0107108082
## D    0    0   27 1901    2 0.0150259067
## E    0    0    2    6 2157 0.0036951501
```

```
#cross validation on my testing data
#out of sample error
predict1 <- predict(modFit, myTest, type = "class")
confusionMatrix(myTest$classe, predict1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    2    0    0    1
##          B   19 1496    3    0    0
##          C    0    4 1362    2    0
##          D    0    0   16 1269    1
##          E    0    0    0    9 1433
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9906, 0.9945)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9915   0.9960   0.9862   0.9914   0.9986
## Specificity            0.9995   0.9965   0.9991   0.9974   0.9986
## Pos Pred Value         0.9987   0.9855   0.9956   0.9868   0.9938
## Neg Pred Value         0.9966   0.9991   0.9971   0.9983   0.9997
## Prevalence             0.2865   0.1914   0.1760   0.1631   0.1829
## Detection Rate         0.2841   0.1907   0.1736   0.1617   0.1826
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9955   0.9963   0.9927   0.9944   0.9986
```

Error

As we can see from the model summaries above, when we run the model on our test data for cross validation we get an accuracy of 99.4% that we can estimate to be our out of sample error. When the model is fitted to the training data used to build the model it shows 100% accuracy, which we can assume as our in sample error.

Apply to final test set

Finally, we apply our model to the final test data. Upon submission all predictions were correct!

```
predict_FINAL <- predict(modFit, test, type = "class")
print(predict_FINAL)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Submission

Prepare the submission:

```
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)
  }
}

pml_write_files(predict_FINAL)
```