Machine Learning Week 4 Project

Yuling Gu

November 10, 2017

Introduction:

The goal of your 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.

GET DATA

```
##load required packages
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##set the working directory
setwd("C:/Users/betty/Desktop")
##import the training and testing dataset
training <- read.csv("pml-training.csv",na.strings = c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv",na.strings = c("NA","#DIV/0!",""))

##get the general idea of the dataset
 dim(training)

## [1] 19622   160

 str(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      : 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 -
0.02 -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]
```

As shown above, the training dataset contains 160 variables,and with some variables contains lots of NA values. We probably want to remove those variables with high volume of NAs, and those non relevent varaibles. I will perform analysis based on only none zero variabels.

CLEANING DATA

```
##remove those near zero variance variables
nzvtrain <- nearZeroVar(training)
training <- training[-nzvtrain]

##remove those variables that does not make sense
training <-training[,7:length(colnames(training))]

##remove those variables that contains high volumes of NAs.
nacol <- as.vector(apply(training,2,function(training)
length(which(!is.na(training)))))
##remove for variables that contains (40%) of NA values.
dropnas <- c()
for (i in 1:length(nacol)) {
     if (nacol[i] > nrow(training)*.40) {
         dropnas <- c(dropnas, colnames(training)[i])
     }
 }
training <- training[,names(training) %in% dropnas]
```

Since we still have a pretty large predictors and datasets, we can break it into 3 seperate dataset to incorportate in 3 different model.

```
set.seed(110)
trainsub <- createDataPartition(training$classe,p=1/3,list=FALSE)
train1 <-training[trainsub,]
temp <- training[-trainsub,]

set.seed(111)
trainsub2 <-createDataPartition(y=temp$classe,p=0.5,list=FALSE)
train2 <- temp[trainsub2,]
```

```
train3 <- temp[-trainsub2,]
dim(train1);dim(train2);dim(train3)

## [1] 6542   53

## [1] 6541   53

## [1] 6539   53
```

Incorportaing with models

1.(gbm) Stochastic boosting trees

2.(rf) random forest decision trees

3.(rpart) decision trees with CART

```
set.seed(112)
mod_fit1 <- train(classe ~ ., method="rf", preProcess=c("center", "scale"),
trControl=trainControl(method = "cv", number = 4), data=train1)

## 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

mod_fit2 <-train(classe~.,method="rpart",preProcess=c("center", "scale"),
trControl=trainControl(method = "cv", number = 4), data=train2)
mod_fit3 <-train(classe~.,method="gbm",preProcess=c("center", "scale"),
trControl=trainControl(method = "cv", number = 4), data=train3,verbose=FALSE)

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

pred_rf <-predict(mod_fit1,train1)
confusionMatrix(pred_rf,train1$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1860    0    0    0    0
##          B    0 1266    0    0    0
##          C    0    0 1141    0    0
##          D    0    0    0 1072    0
##          E    0    0    0    0 1203
## 
## Overall Statistics
## 
##                Accuracy : 1          
##                  95% CI : (0.9994, 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   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
pred_rpart <-predict(mod_fit2,train2)
confusionMatrix(pred_rpart,train2$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1854 1266 1141 1072  650
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    6    0    0    0  552
## 
## Overall Statistics
## 
##                Accuracy : 0.3678          
##                  95% CI : (0.3561, 0.3797)
##     No Information Rate : 0.2844          
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1271
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.9968   0.0000   0.0000   0.0000  0.45923
## Specificity             0.1179   1.0000   1.0000   1.0000  0.99888
## Pos Pred Value          0.3099      NaN      NaN      NaN  0.98925
## Neg Pred Value          0.9892   0.8065   0.8256   0.8361  0.89136
## Prevalence              0.2844   0.1935   0.1744   0.1639  0.18376
## Detection Rate          0.2834   0.0000   0.0000   0.0000  0.08439
## Detection Prevalence    0.9147   0.0000   0.0000   0.0000  0.08531
## Balanced Accuracy       0.5573   0.5000   0.5000   0.5000  0.72906
```

```
pred_gbm <-predict(mod_fit3,train3)
confusionMatrix(pred_gbm,train3$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1852   27    0    0    0
##          B    4 1225   18    3    5
##          C    3   13 1115   25    6
##          D    0    0    6 1042   10
##          E    1    0    1    2 1181
##
## Overall Statistics
##
##                Accuracy : 0.981
##                  95% CI : (0.9774, 0.9842)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.9957   0.9684   0.9781   0.9720   0.9825
## Specificity             0.9942   0.9943   0.9913   0.9971   0.9993
## Pos Pred Value          0.9856   0.9761   0.9596   0.9849   0.9966
## Neg Pred Value          0.9983   0.9924   0.9954   0.9945   0.9961
## Prevalence              0.2844   0.1935   0.1743   0.1639   0.1838
## Detection Rate          0.2832   0.1873   0.1705   0.1594   0.1806
## Detection Prevalence   0.2874   0.1919   0.1777   0.1618   0.1812
## Balanced Accuracy      0.9950   0.9813   0.9847   0.9845   0.9909
```

Based on those three model, random forest have the most acctuary. So I decide to accept the random forest model as the champion and move on to prediction in the testing sample.

```
print(predTest <-predict(mod_fit1,newdata=testing))

##  [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
```