Prediction_Assignment_Writeup

2015-09-27

```
library(caret)
library(rpart)
library(rattle)
library(randomForest)
library(knitr)
```

Project

Project Introduction

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).

Data

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.

Goal

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.

Getting And Loading The Data

```
set.seed(12345)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

colnames_train <- colnames(training)
```

Cleaning The Data

The first step was to clean the data from all kind of missing values and columns that may be irrelvant to prediction (i.e - near zero variance columns) ###Removing Columns With NAs

```
# Count the number of non-NAs in each col.
nonNAs <- function(x) {
    as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}

# Build vector of missing data or NA columns to drop.
colcnts <- nonNAs(training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}
```

Removing Irrelevant Columns

These columns contains irrelvant information for the prediction algorithem.

```
# Drop NA data and the first 7 columns as they're unnecessary for predicting.
training <- training[,!(names(training) %in% drops)]
training <- training[,8:length(colnames(training))]
```

```
testing <- testing[,!(names(testing) %in% drops)]
testing <- testing[,8:length(colnames(testing))]
```

Show Remaining Columns Training vs. Testing

```
# Show remaining columns training.
colnames(training)

##  [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"     "classe"

# Show remaining columns testing
colnames(testing)

##  [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"     "problem_id"
```

Removing Columns With Near Zero Variance

```r
print(nearZeroVar(training, saveMetrics=TRUE))
```

```
##                        freqRatio percentUnique zeroVar   nzv
## roll_belt               1.101904     6.7781062   FALSE FALSE
## pitch_belt              1.036082     9.3772296   FALSE FALSE
## yaw_belt                1.058480     9.9734991   FALSE FALSE
## total_accel_belt        1.063160     0.1477933   FALSE FALSE
## gyros_belt_x            1.058651     0.7134849   FALSE FALSE
## gyros_belt_y            1.144000     0.3516461   FALSE FALSE
## gyros_belt_z            1.066214     0.8612782   FALSE FALSE
## accel_belt_x            1.055412     0.8357966   FALSE FALSE
## accel_belt_y            1.113725     0.7287738   FALSE FALSE
## accel_belt_z            1.078767     1.5237998   FALSE FALSE
## magnet_belt_x           1.090141     1.6664968   FALSE FALSE
## magnet_belt_y           1.099688     1.5187035   FALSE FALSE
## magnet_belt_z           1.006369     2.3290184   FALSE FALSE
## roll_arm               52.338462    13.5256345   FALSE FALSE
## pitch_arm              87.256410    15.7323412   FALSE FALSE
## yaw_arm                33.029126    14.6570176   FALSE FALSE
## total_accel_arm         1.024526     0.3363572   FALSE FALSE
## gyros_arm_x             1.015504     3.2769341   FALSE FALSE
## gyros_arm_y             1.454369     1.9162165   FALSE FALSE
## gyros_arm_z             1.110687     1.2638875   FALSE FALSE
## accel_arm_x             1.017341     3.9598410   FALSE FALSE
## accel_arm_y             1.140187     2.7367241   FALSE FALSE
## accel_arm_z             1.128000     4.0362858   FALSE FALSE
## magnet_arm_x            1.000000     6.8239731   FALSE FALSE
## magnet_arm_y            1.056818     4.4439914   FALSE FALSE
## magnet_arm_z            1.036364     6.4468454   FALSE FALSE
## roll_dumbbell           1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell          2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell            1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell    1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x        1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y        1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z        1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x        1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y        1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z        1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x       1.098266     5.7486495   FALSE FALSE
```

```
## magnet_dumbbell_y      1.197740     4.3012945   FALSE  FALSE
## magnet_dumbbell_z      1.020833     3.4451126   FALSE  FALSE
## roll_forearm          11.589286    11.0895933   FALSE  FALSE
## pitch_forearm         65.983051    14.8557741   FALSE  FALSE
## yaw_forearm           15.322835    10.1467740   FALSE  FALSE
## total_accel_forearm    1.128928     0.3567424   FALSE  FALSE
## gyros_forearm_x        1.059273     1.5187035   FALSE  FALSE
## gyros_forearm_y        1.036554     3.7763735   FALSE  FALSE
## gyros_forearm_z        1.122917     1.5645704   FALSE  FALSE
## accel_forearm_x        1.126437     4.0464784   FALSE  FALSE
## accel_forearm_y        1.059406     5.1116094   FALSE  FALSE
## accel_forearm_z        1.006250     2.9558659   FALSE  FALSE
## magnet_forearm_x       1.012346     7.7667924   FALSE  FALSE
## magnet_forearm_y       1.246914     9.5403119   FALSE  FALSE
## magnet_forearm_z       1.000000     8.5771073   FALSE  FALSE
## classe                 1.469581     0.0254816   FALSE  FALSE
```

No headers with nzr were found.

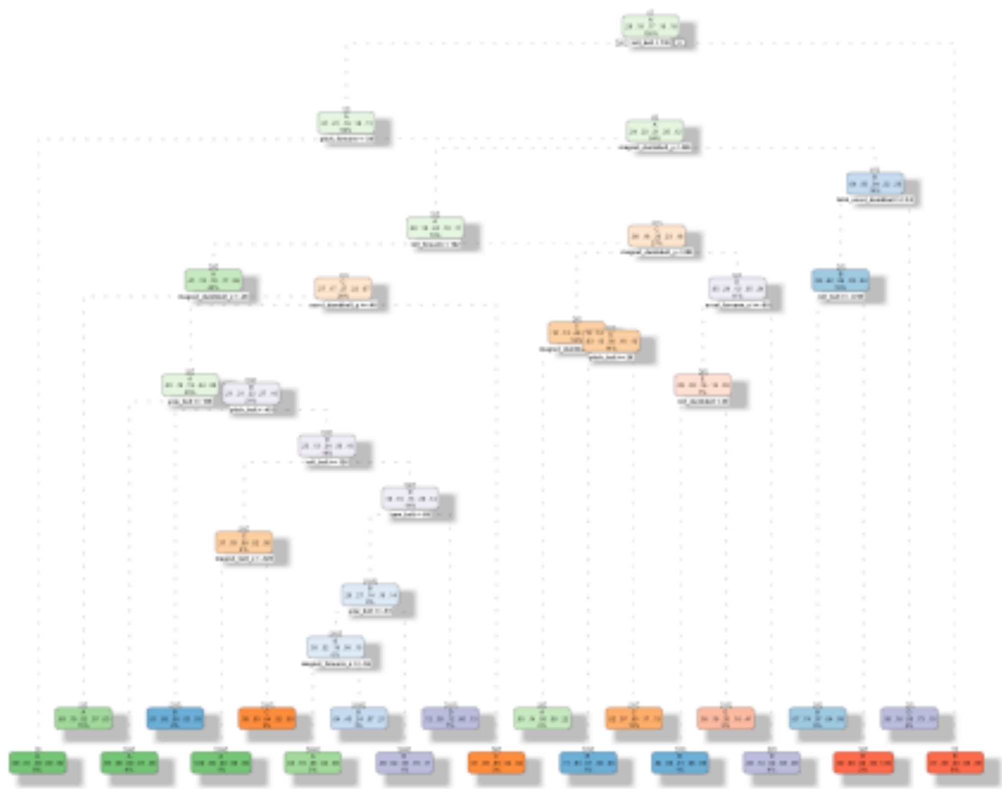
Partioning The Training Set

```
set.seed(666)
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
df_training <- training[inTrain,]
df_testing <- training[-inTrain,]
```

Prediction With Decision Tree

```
modFitA <- rpart(classe ~ ., data=df_training, method="class")
fancyRpartPlot(modFitA)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some
overplotting
```



Rattle 2015-Sep-27 16:48:24 okatz

```
predictionsA <- predict(modFitA, df_testing, type = "class")
confusionMatrix(predictionsA, df_testing$classe)
```

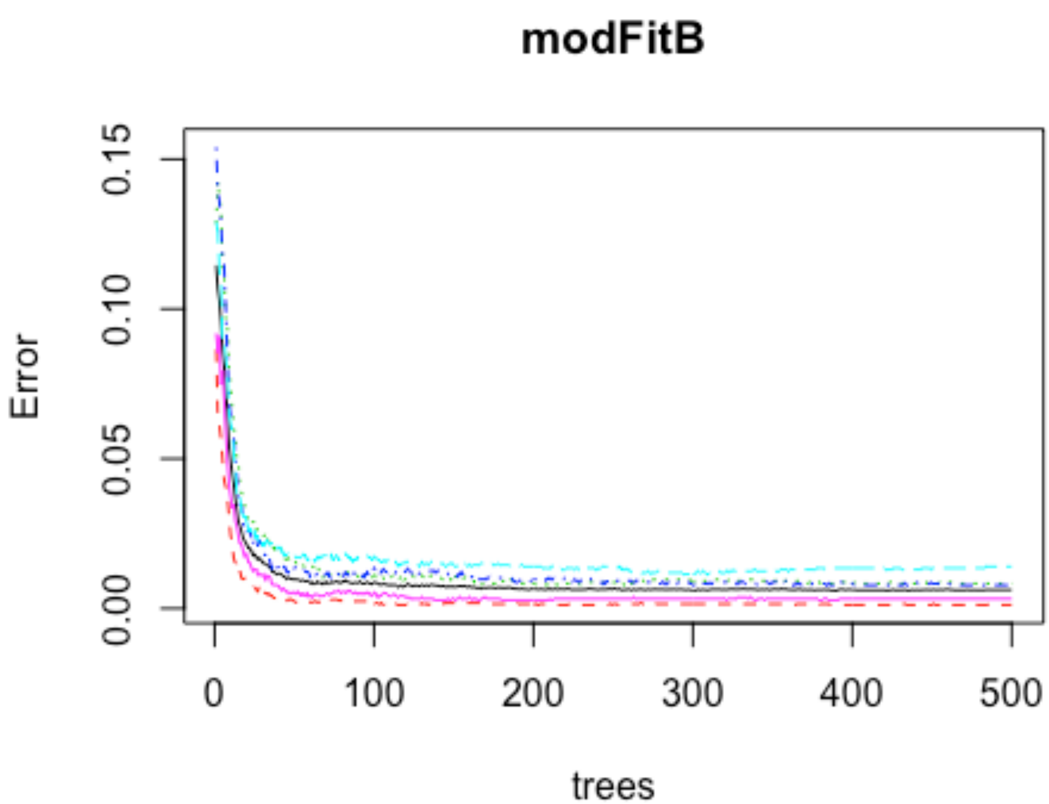
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1970  344   44  131   99
##          B   85  890  140   49  133
##          C   45  106 1001  178  132
##          D  105  100   99  856  142
##          E   27   78   84   72  936
##
## Overall Statistics
##
##                Accuracy : 0.7205
##                  95% CI : (0.7104, 0.7304)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6446
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8826   0.5863   0.7317   0.6656   0.6491
## Specificity            0.8899   0.9357   0.9288   0.9320   0.9592
## Pos Pred Value         0.7612   0.6862   0.6847   0.6575   0.7820
## Neg Pred Value         0.9502   0.9041   0.9425   0.9343   0.9239
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2511   0.1134   0.1276   0.1091   0.1193
## Detection Prevalence   0.3298   0.1653   0.1863   0.1659   0.1526
## Balanced Accuracy      0.8863   0.7610   0.8303   0.7988   0.8042
```

The accuracy is not good enough 72% therefore I also tried Random forest algo to see if we can find better prediction.

Prediction With Random Forest

```
set.seed(666)
modFitB <- randomForest(classe ~. , data=df_training)
plot(modFitB)
```




```
predictionsB <- predict(modFitB, df_testing, type = "class")
confusionMatrix(predictionsB, df_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 2229   12    0    0    0
##          B    3 1500   20    0    0
##          C    0    6 1345   14    0
##          D    0    0    3 1272    5
##          E    0    0    0    0 1437
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9897, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9898
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9881   0.9832   0.9891   0.9965
## Specificity            0.9979   0.9964   0.9969   0.9988   1.0000
## Pos Pred Value         0.9946   0.9849   0.9853   0.9938   1.0000
## Neg Pred Value         0.9995   0.9972   0.9965   0.9979   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1912   0.1714   0.1621   0.1832
## Detection Prevalence   0.2856   0.1941   0.1740   0.1631   0.1832
## Balanced Accuracy      0.9983   0.9923   0.9900   0.9939   0.9983
```

Random forest yield better results with 99% accuracy!

Assignment Submission & Result Prediction

Random Forests gave an Accuracy on the training dataset of 99.2%, which was more accurate that what I got from the Decision Tree with 72.05%. The expected out-of-sample error is 100-99.2 = 0.8%.

```
predictionsTest <- predict(modFitB, testing, type = "class")

predictionsTest

##  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

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=FA
LSE)
    }
}

pml_write_files(predictionsTest)
```