Practical Machine Learning Course Project

Angela

4/8/2020

Project Overview

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 dataset, six male participants aged between 20-28 years, with little weight lifting experience, were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions.

- Each class is defined as:
  - exactly according to the specification (Class A)
  - throwing the elbows to the front (Class B)
  - lifting the dumbbell only halfway (Class C)
  - lowering the dumbbell only halfway (Class D)
  - throwing the hips to the front (Class E)

The goal of your project is to predict the manner in which they did the exercise. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

- You should create a report describing:
  - how you built your model
  - how you used cross validation
  - what you think the expected out of sample

Outline of report

1. Load packages
2. Load data
3. Clean data
4. Separate data into testing and training datasets
5. Create basic decision tree model and test
6. Create random forest model and test

1. Load all relevant packages

This includes the caret, rattle, randomForest, randomForestExplainer and cowplot packages. See versions used in Appendix - Figure 9.

2. Load data

```
url <- "http://groupware.les.inf.puc-rio.br/static/WLE/WearableComputing_weight_lifting_exercises_biceps
df <- read.csv(url, header = TRUE, na.strings=c("NA","#DIV/0!",""))
```

3. Clean up data

```
#Remove columns containing ALL NA values
df <- df[,colSums(is.na(df)) == 0]

#Remove all irrelevant columns that you will not need as predictors
df <- subset(df, select = -c(1:7))
```

Separate data into testing and training datasets

The links provided for the class did not include the classe variable so my training and tests datasets will look slightly different as I took them from the main source and re-separated them.
training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
testing: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
main source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har
See Figure 1 and 2 for more information on data.

```
#Create training and testing datasets
inTrain <- createDataPartition(y = df$classe,
                               p=0.7, list = FALSE)
training <- df[inTrain,]
testing <- df[-inTrain,]
```

4. Create basic decision tree model and test

The basic decision tree model is only composed of one tree where the classe variable is the outcome and there are 51 predictors, which are listed in Figure 1. This model gives us a 48% accuracy on our training set (Figure 3) and a 47.9% accurancy on our test set (Figure 4). Many variables are miscategorized as you can see by looking at the confusion matrices (Figure 3-5). Therefore, we will improve this model by using a random forest.

```
set.seed(1234)

#Make simple decision tree model using rpart
model <- train(classe ~ ., data = training, method="rpart")

#Apply model to training dataset and see accuracy
rpartmodel_train <- predict(model, data = training)
CM_train <- confusionMatrix(training$classe, rpartmodel_train)

#Apply model to testing dataset and see accuracy
rpartmodel_test <- predict(model, newdata = testing)
rpart_CM_test <- confusionMatrix(testing$classe, rpartmodel_test)
```

5. Create random forest model and test

In order to improve our accuracy, we will use a random forest model. This model generates 100 decision trees (ntree) where each node splits based off of 7 variables (mtry) and gives us variable importance based on majority voting. Using random forest, both our training and testing datasets have an accuracy of >99% with a very small amount of misclassification (Figure 6-7). These can be visualized using multiway importance plots and variable importance plots (Figure 8).

```
set.seed(2020)

#Generate random forest model
rfmodel <- randomForest(classe ~ ., data = training, method="rf", ntree=100, importance = TRUE)

#See how well the random forest model performs on training dataset
rfmodel_train <- predict(rfmodel, data = training)
rf_CM_train <- confusionMatrix(training$classe, rfmodel_train)

#See which variables are the most important in the training dataset
variableimportance <- importance(rfmodel)
variableimportance <- varImp(rfmodel)[1:10,]

min_depth_frame <- min_depth_distribution(rfmodel)
plot1 <- plot_min_depth_distribution(min_depth_frame)

importance_frame <- measure_importance(rfmodel)
plot2 <- plot_multi_way_importance(importance_frame, size_measure = "no_of_nodes")

#See how well the random forest model performs on testing dataset
rfmodel_test <- predict(rfmodel, newdata = testing)
rf_CM_test <- confusionMatrix(testing$classe, rfmodel_test)
```

For our Random Forest model, our error rate is 0.17%.

6. Conclusions

Based on both our simple decision tree and random forest model, the roll_belt, pitch_forearm and magnet_dumbbell_y variables contribute the most importance to our model. Therefore, we hypothesize that by looking at these variables (and other in the top ten from random forest) one should be able to predict common workout errors.

7. Appendix

Figure 1. Name of all variables included in model

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

Figure 2. Dimensions of training and testing datasets

These datasets were split 70:30 where the training set had more data.

```
## [1] 27472    51
```

```
## [1] 11770    51
```

Figure 3. Rpart training information

```
##    rpartmodel_train
##        A    B    C    D    E
##   A 7072  183  533    1   23
##   B 2128 1751 1258  178    1
##   C 2203  170 2414    4    0
##   D 1985  103 1612  803    0
##   E 1128  849 1394  235 1444
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.4908270      0.3347058      0.4848985      0.4967575      0.5283925
## AccuracyPValue  McnemarPValue
##      1.0000000      0.0000000
```

Figure 4. Rpart testing information

```
##    rpartmodel_test
##        A    B    C    D    E
##   A 3012   75  256    0    4
##   B  905  690  609   72    1
##   C  943   79 1029    2    0
##   D  840   50  710  329    0
##   E  467  347  622  102  626
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.4830926      0.3251059      0.4740275      0.4921661      0.5239592
## AccuracyPValue  McnemarPValue
```

```
##       1.0000000       0.0000000
```

Figure 5. Rpart visualization

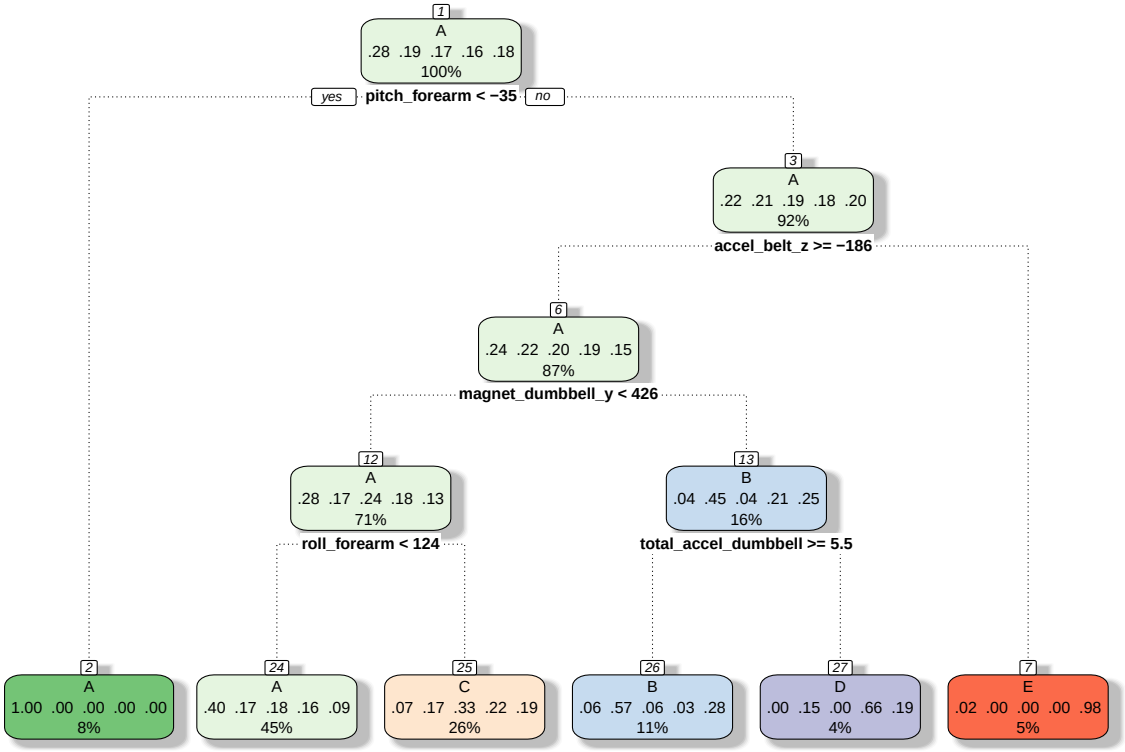


Figure 6. Random forest information - training

```
##    rfmodel_train
##        A    B    C    D    E
##   A 7810    0    1    0    1
##   B   10 5305    1    0    0
##   C    0    5 4783    3    0
##   D    0    0   19 4481    3
##   E    0    0    2    7 5041
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9981072      0.9976058      0.9975185      0.9985860      0.2846535
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 7810    0    1    0    1
##          B   10 5305    1    0    0
##          C    0    5 4783    3    0
##          D    0    0   19 4481    3
##          E    0    0    2    7 5041
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9975, 0.9986)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9991   0.9952   0.9978   0.9992
## Specificity            0.9999   0.9995   0.9996   0.9990   0.9996
## Pos Pred Value         0.9997   0.9979   0.9983   0.9951   0.9982
## Neg Pred Value         0.9995   0.9998   0.9990   0.9996   0.9998
## Prevalence             0.2847   0.1933   0.1749   0.1635   0.1836
## Detection Rate         0.2843   0.1931   0.1741   0.1631   0.1835
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9993   0.9993   0.9974   0.9984   0.9994
```

```
##                            A         B         C         D         E
## pitch_belt       15.902882 23.183642 19.633393 19.569415 19.099636
## yaw_belt         34.441314 26.280667 22.033612 26.547584 17.263394
## total_accel_belt  7.608868 10.335657  7.273549  7.652050  8.710931
## gyros_belt_x      8.181354 11.939085 10.896166  9.476317 11.402628
## gyros_belt_y      5.232421  7.546172  9.173653  7.334043 11.456449
## gyros_belt_z     12.088190 16.608631 13.037057 13.214744 15.206548
## accel_belt_x      7.881268  9.421311  8.782041  7.404083  9.145018
## accel_belt_y      6.925048  9.065617  9.718086  8.871146  9.251272
## accel_belt_z     10.599258 13.618333 14.872851 12.116033 10.585698
## magnet_belt_x     8.574250 14.739932 13.469375 10.552141 18.943541
```

Figure 7. Random forest information - testing

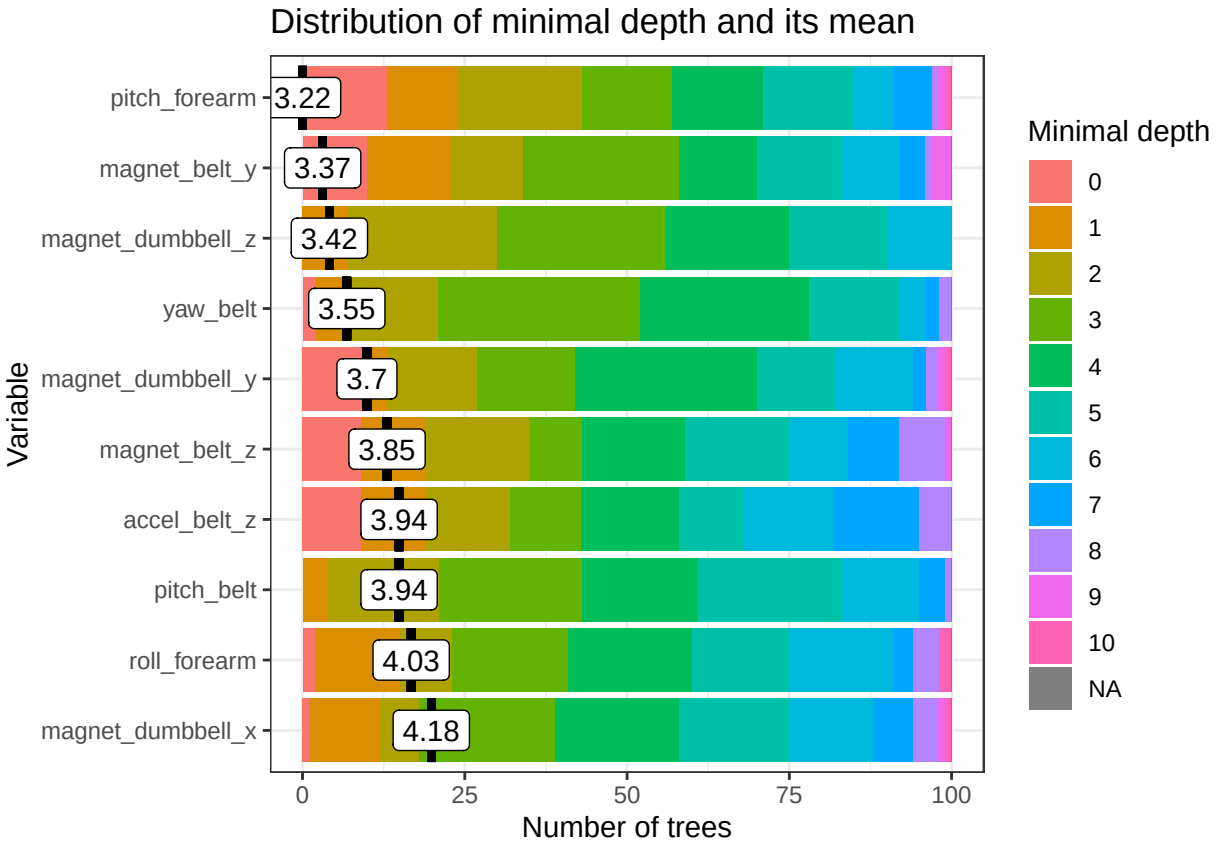
```
##    rfmodel_test
##        A    B    C    D    E
##   A 3347    0    0    0    0
##   B    0 2277    0    0    0
##   C    0    4 2049    0    0
##   D    0    0    4 1925    0
##   E    0    0    0    4 2160
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9989805      0.9987104      0.9982197      0.9994731      0.2843670
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3347    0    0    0    0
##          B    0 2277    0    0    0
```

```
##          C    0    4 2049    0    0
##          D    0    0    4 1925    0
##          E    0    0    0    4 2160
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9982, 0.9995)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9981   0.9979   1.0000
## Specificity            1.0000   1.0000   0.9996   0.9996   0.9996
## Pos Pred Value         1.0000   1.0000   0.9981   0.9979   0.9982
## Neg Pred Value         1.0000   0.9996   0.9996   0.9996   1.0000
## Prevalence             0.2844   0.1938   0.1744   0.1639   0.1835
## Detection Rate         0.2844   0.1935   0.1741   0.1636   0.1835
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   0.9991   0.9988   0.9988   0.9998
```

Figure 8. Random forest visualization


Distribution of minimal depth and its mean
pitch_forearm
magnet_belt_y
magnet_dumbbell_z
yaw_belt
magnet_dumbbell_y
magnet_belt_z
accel_belt_z
pitch_belt
roll_forearm
magnet_dumbbell_x
3.22
3.37
3.42
3.55
3.7
3.85
3.94
3.94
4.03
4.18
Variable
0
25
50
75
100
Number of trees
Minimal depth
0
1
2
3
4
5
6
7
8
9
10
NA

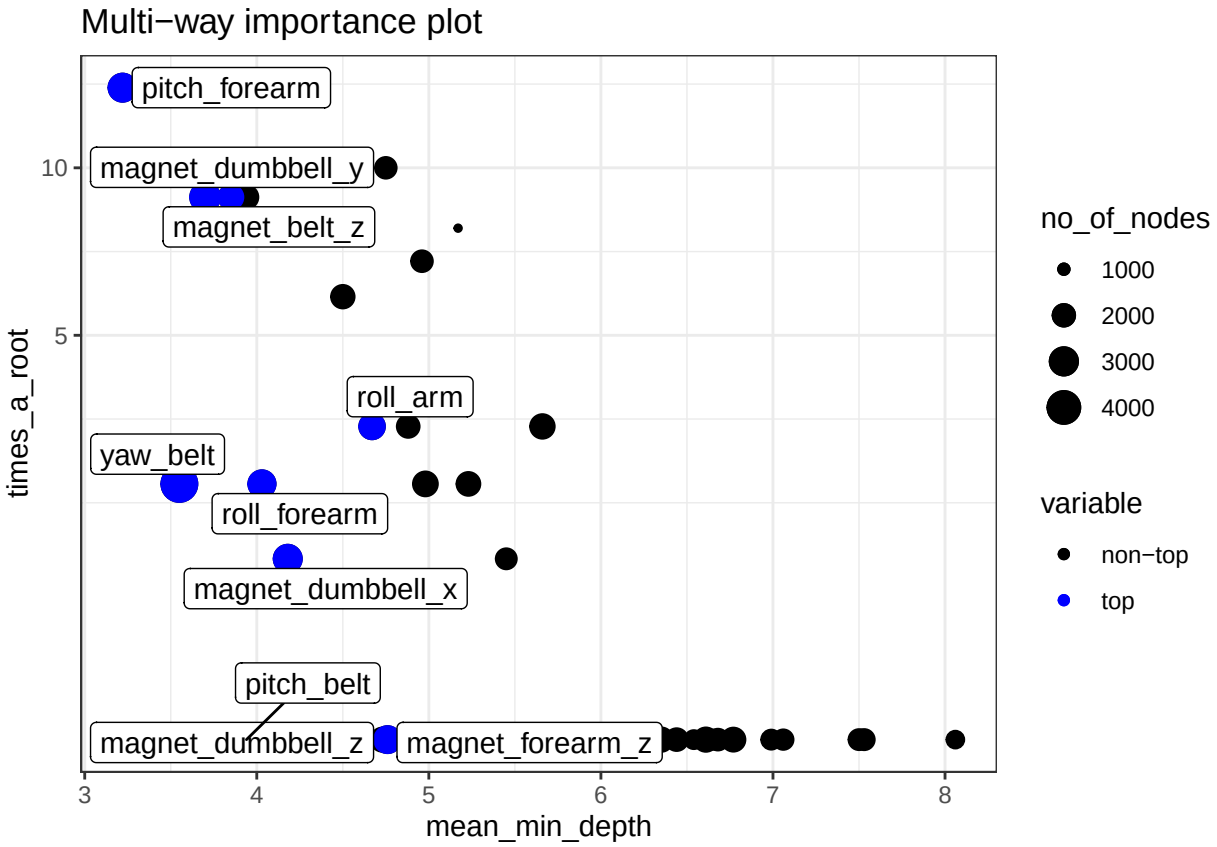

Multi−way importance plot
pitch_forearm
magnet_dumbbell_y
magnet_belt_z
roll_arm
yaw_belt
roll_forearm
magnet_dumbbell_x
pitch_belt
magnet_dumbbell_z
magnet_forearm_z
10
5
times_a_root
3
4
5
6
7
8
mean_min_depth
no_of_nodes
1000
2000
3000
4000
variable
non−top
top

Figure 9. List of all packages used and what version

```
version$version.string
```

```
## [1] "R version 3.5.3 (2019-03-11)"
```

```
packageVersion("caret", lib.loc = NULL)
```

```
## [1] '6.0.85'
```

```
packageVersion("rattle", lib.loc = NULL)
```

```
## [1] '5.3.0'
```

```
packageVersion("randomForest", lib.loc = NULL)
```

```
## [1] '4.6.14'
```

```
packageVersion("randomForestExplainer", lib.loc = NULL)
```

```
## [1] '0.10.0'
```

```
packageVersion("cowplot", lib.loc = NULL)
```

```
## [1] '1.0.0'
```