Peer-graded Assignment Prediction Assignment Writeup

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: link

(see the section on the Weight Lifting Exercise Dataset).

Data

The **training** data for this project are available here:

training data

The **test** data are available here:

testing data

The data for this project come from this source: Groupware. The data used in this project has been generously provided by the authors, Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. *"Qualitative Activity Recognition of Weight Lifting Exercises."*" Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Project Goal

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, the main goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

The goal of the 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.

Loading libraries

```
library(ggplot2)
library(caret)
```

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

```
library(randomForest)
```

```
## 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
```

```
library(e1071)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(rpart)
library(rpart.plot)
library(plyr)
```

Loading data

```
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

The training dataset has 19622 observations and 160 variables, and the testing data set contains 20 observations and the same variables as the training set.

NA's will be removed and we select the columns we need for the analysis

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

Data splitting

we proceed with partition rows into training and crossvalidation

```
training$classe = factor(training$classe)
inTrain <- createDataPartition(training$classe, p = 0.75, list=FALSE)
training <- training[inTrain, ]
crossval <- training[-inTrain, ]
dim(training)
```

```
## [1] 14718    53
```

Modelling

We'll use the train function, methods: rpart and rf to find the best prediction.

Mod method: rpart

```r
control <- trainControl(method = "cv", number = 5)
modelFit_rpart <- train(classe ~ ., data = training, method = "rpart", trControl = control)
print(modelFit_rpart, digits = 4)
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11776, 11776, 11772, 11774, 11774
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03427  0.5190    0.3722
##   0.06060  0.4412    0.2505
##   0.11383  0.3484    0.0976
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03427.
```

Prediction and Confusion Matrix

```r
predict_rpart <- predict(modelFit_rpart, crossval)
# Show prediction result
confusionMatrix(crossval$classe, predict_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 942  17  98   0   4
##          B 299 239 155   0   0
##          C 312  15 314   0   0
##          D 259 114 235   0   0
##          E 102 106 191   0 292
##
## Overall Statistics
##
##                Accuracy : 0.4838
##                  95% CI : (0.4675, 0.5)
##     No Information Rate : 0.5181
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3248
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4922   0.4868   0.3162       NA  0.98649
```

```
## Specificity            0.9331   0.8583   0.8789   0.8354  0.88258
## Pos Pred Value         0.8878   0.3449   0.4899       NA  0.42258
## Neg Pred Value         0.6308   0.9160   0.7776       NA  0.99867
## Prevalence             0.5181   0.1329   0.2688   0.0000  0.08013
## Detection Rate         0.2550   0.0647   0.0850   0.0000  0.07905
## Detection Prevalence   0.2872   0.1876   0.1735   0.1646  0.18706
## Balanced Accuracy      0.7127   0.6725   0.5976       NA  0.93453
```

From the confusion matrix, the accuracy rate is 0.48, and so the out-of-sample error rate is 0.5. Using classification tree does not predict the outcome classe very well.

Mod method: rf (randomForest)

```
rf_modelfit <- randomForest(classe ~ ., data=training)
print(rf_modelfit, digits = 4)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.48%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4179    5    0    0    1 0.001433692
## B   15 2831    2    0    0 0.005969101
## C    0   12 2554    1    0 0.005064277
## D    0    0   22 2387    3 0.010364842
## E    0    0    4    6 2696 0.003695492
```

Prediction and Confusion Matrix

```
rf_predict <- predict(rf_modelfit, crossval)
confusionMatrix(rf_predict, crossval$classe)
```

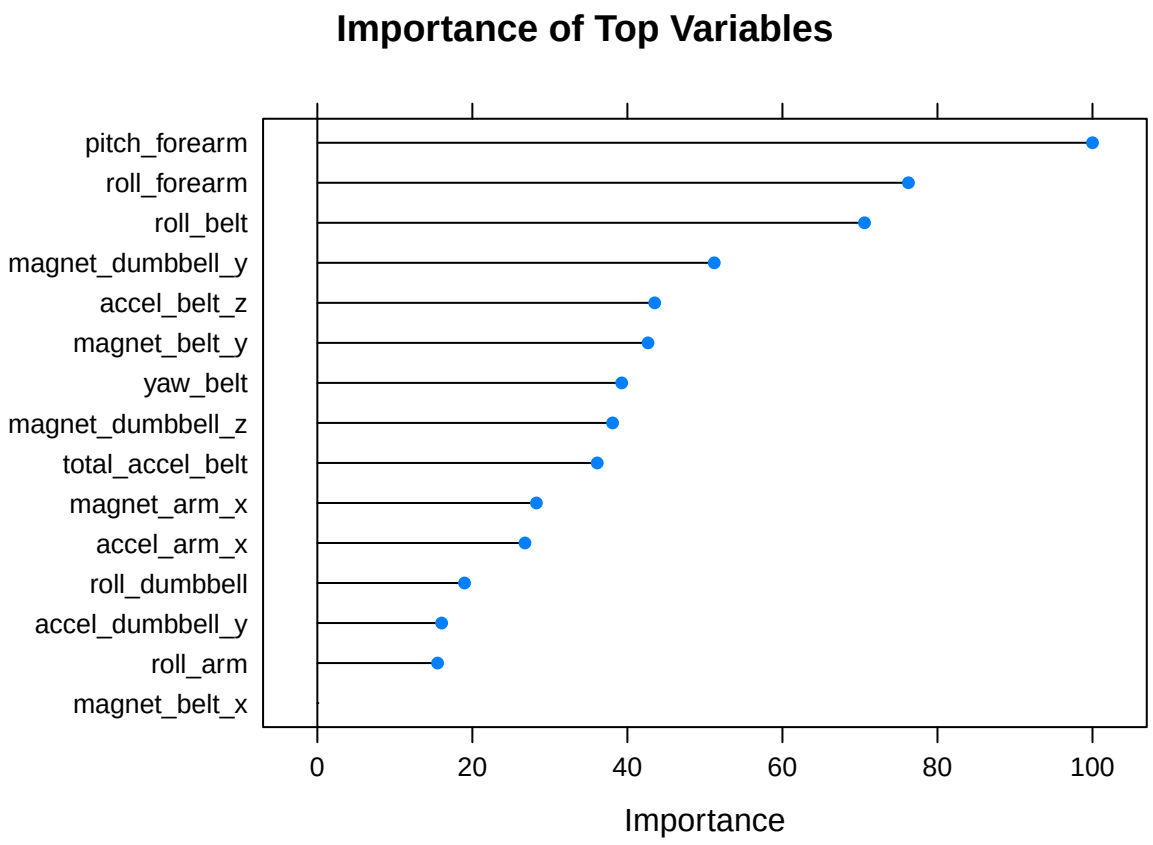
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061    0    0    0    0
##          B    0  693    0    0    0
##          C    0    0  641    0    0
##          D    0    0    0  608    0
##          E    0    0    0    0  691
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2872
##     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             0.2872   0.1876   0.1735   0.1646   0.1871
## Detection Rate         0.2872   0.1876   0.1735   0.1646   0.1871
## Detection Prevalence   0.2872   0.1876   0.1735   0.1646   0.1871
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

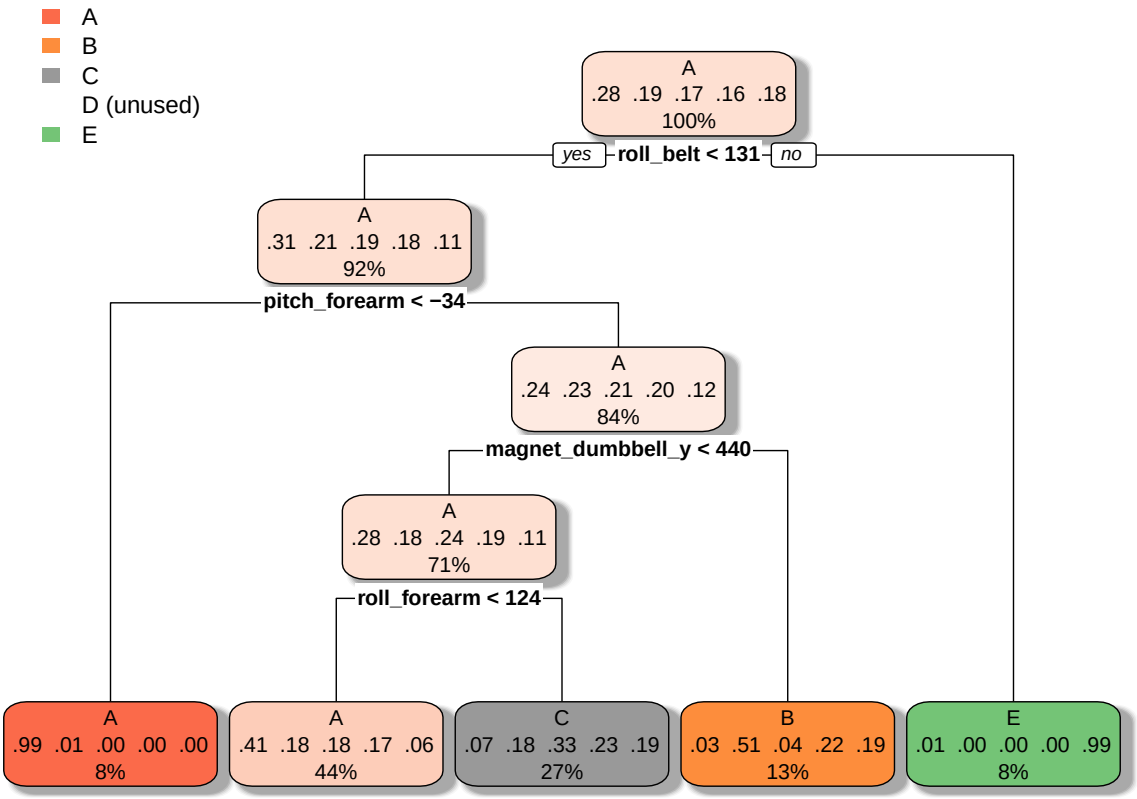
This model achieved 100% accuracy on the validation set. Meaning **random forest method** is way better than classification tree method.

Plotting Fine tuning & trees

```
modelFit_rpartOBJ <- varImp(modelFit_rpart)
plot(modelFit_rpartOBJ, main = "Importance of Top Variables", top = 15)
```




```
rpart.plot(modelFit_rpart$finalModel, shadow.col="darkgray")
```



Prediction on Testing data

With the above conlusion, We now use random forests to predict the outcome variable classe for the testing set.

```
(predict(rf_modelfit, testing))
```

```
##  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
```

Conclusion

With the **Random forest algorithm**, the 20 predicted values are as shown above.

Submition Date: 24/03/2021