Week 4 PML assignment

Palvinder Kaur

4 November 2019

##1. 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).

##2. Data Sources 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.

##3. Load & Prep Data

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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:rattle':
##
##     importance

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(RColorBrewer)
```

```r
set.seed(12345)
```

```r
url_trainset <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_testset  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```r
train <- read.csv(url(url_trainset), strip.white = TRUE, na.strings=c("NA",""))
test  <- read.csv(url(url_testset),  strip.white = TRUE, na.strings = c("NA",""))
```

```r
dim(train)
```

```
## [1] 19622   160
```

```r
dim(test)
```

```
## [1]  20 160
```

remove NA and near zero variables

```r
 train <- train[, colSums(is.na(train)) == 0]
 test <- test[, colSums(is.na(test)) == 0]
 classe <- train$classe
 trainR <- grepl("^X|timestamp|window", names(train))
 train <- train[, !trainR]
 trainM <- train[, sapply(train, is.numeric)]
 trainM$classe <- classe
 testR <- grepl("^X|timestamp|window", names(test))
 test<- test[, !testR]
 testM <- test[, sapply(test, is.numeric)]
```

```r
dim(trainM)
```

```
## [1] 19622    53
```

```r
dim(testM)
```

```
## [1] 20 53
```

#4. Partition Data -70% train; 30% test

```r
inTrain <- createDataPartition(trainM$classe, p=0.70, list=F)
train_data <- trainM[inTrain, ]
test_data <- trainM[-inTrain, ]
```

#5. Build Random Forest model & check performance

```r
setting_rf <- trainControl(method="cv", 5)
rf <- train(classe ~ ., data=train_data, method="rf", trControl=setting_rf, ntree=250)
rf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10987, 10990, 10990, 10991, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9908273  0.9883962
##   27    0.9903908  0.9878444
##   52    0.9838396  0.9795564
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

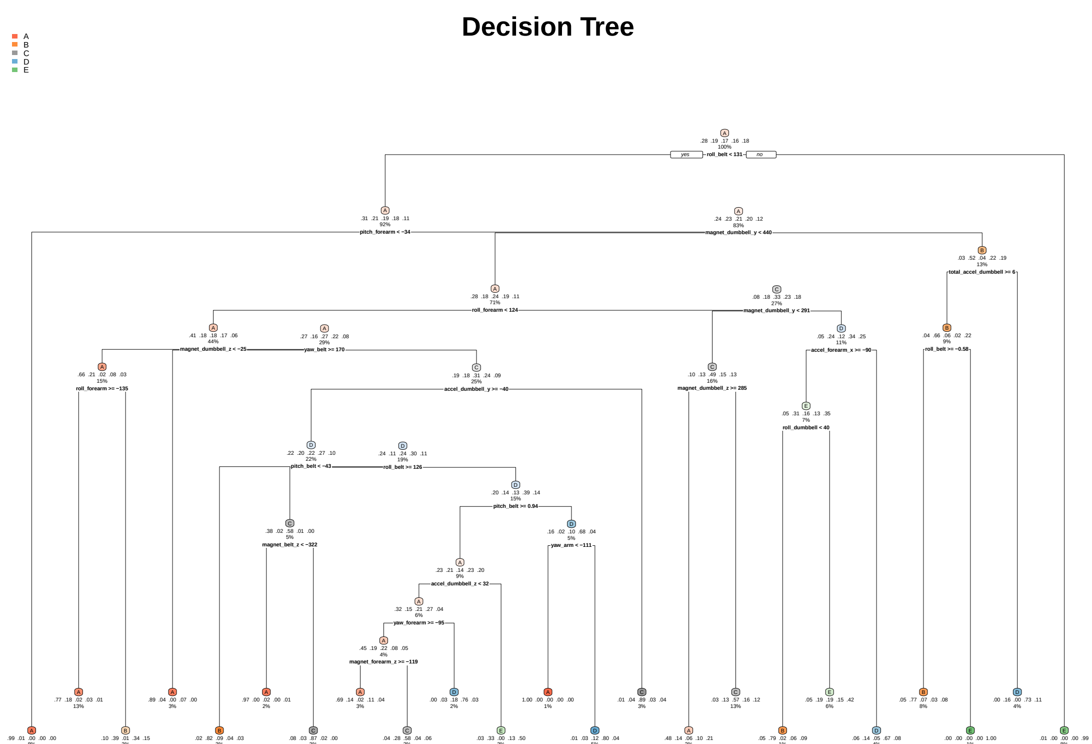
```r
#check rf model performance
predict_rf <- predict(rf, test_data)
confusionMatrix(test_data$classe, predict_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    6 1133    0    0    0
##          C    0    7 1018    1    0
##          D    0    0   23  940    1
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9930   0.9779   0.9989   0.9991
## Specificity            0.9998   0.9987   0.9983   0.9951   1.0000
## Pos Pred Value         0.9994   0.9947   0.9922   0.9751   1.0000
## Neg Pred Value         0.9986   0.9983   0.9953   0.9998   0.9998
## Prevalence             0.2853   0.1939   0.1769   0.1599   0.1840
## Detection Rate         0.2843   0.1925   0.1730   0.1597   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9959   0.9881   0.9970   0.9995
```

#6. Build decision tree & check performance

```r
dt_model <- rpart(classe ~ ., data = train_data, method = "class")
dt_predict <- predict(dt_model, test_data, type = "class")
rpart.plot(dt_model, main = "Decision Tree", under = T, faclen = 0)
```




```r
#check dt model performance
confusionMatrix(dt_predict, test_data$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1532  176   28   48   41
##          B   54  585   57   64   76
##          C   35  154  819  134  126
##          D   25   76   58  631   56
##          E   28  148   64   87  783
##
## Overall Statistics
##
##                Accuracy : 0.7392
##                  95% CI : (0.7277, 0.7503)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6692
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9152  0.51361   0.7982   0.6546   0.7237
## Specificity            0.9304  0.94711   0.9076   0.9563   0.9319
## Pos Pred Value         0.8395  0.69976   0.6459   0.7459   0.7054
## Neg Pred Value         0.9650  0.89028   0.9552   0.9339   0.9374
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2603  0.09941   0.1392   0.1072   0.1331
## Detection Prevalence   0.3101  0.14206   0.2155   0.1438   0.1886
## Balanced Accuracy      0.9228  0.73036   0.8529   0.8054   0.8278
```

#7. Conclusion

Random forest tree performed better than the decision tree in terms of model statistics (accuracy: 99.5% vs 73.9%). Out of sample rate for the rf model was estimated to be 0.05%