Practical Machine Learning - final project

Francesco Andreini

31 July 2018

Introduction: data retrieval and cleaning

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, the 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 (http://groupware.les.inf.puc-rio.br/har), to which we are grateful for the data available.

Firstly, we load all the libraries we need and afterwe retrieve data for training and data set. After that, we partition the training data in training and test set. The testing data will be used as validation set.

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.4.4
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## 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
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.4
```

```
set.seed(12345)
pml_training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=T)
validation <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=T)

Training <- createDataPartition(pml_training$classe, p=0.7, list=FALSE)
train <- pml_training[Training, ]
test <- pml_training[-Training, ]

dim(train)
```

```
## [1] 13737   160
```

```
dim(test)
```

```
## [1] 5885  160
```

```
dim(validation)
```

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

We then remove (near) zero variance predictors and/or predictors that are have both of the following characteristics: very few unique values relative to the number of samples and ratio of the frequency of the most common value to the frequency of the second most common value very large.

```
nzv <- nearZeroVar(train, saveMetrics=TRUE)
train <- train[,nzv$nzv==FALSE]
test <- test[,nzv$nzv==FALSE]
validation <- validation[,nzv$nzv==FALSE]
train<-train[,-1]
test<-test[,-1]
validation<-validation[,-1]

dim(train)
```

```
## [1] 13737   105
```

```
dim(test)
```

```
## [1] 5885  105
```

```
dim(validation)
```

```
## [1]  20 105
```

Finally, we delete all columns with more than 70% of missing values from the training, the test and the validation sets.

```
columns_to_excl<-c()
for (x in 1:ncol(train)){
  if (sum(is.na(train[,x]))/nrow(train)<=0.7 || colnames(train)[x]=='classe'){
    columns_to_excl<-c(columns_to_excl,x)
  }
}
train<-train[,columns_to_excl]
test<-test[,columns_to_excl]
validation<-validation[,columns_to_excl]

dim(train)
```

```
## [1] 13737    58
```

```
dim(test)
```

```
## [1] 5885   58
```

```
dim(validation)
```

```
## [1] 20 58
```

Predictions

Random Forests

We firstly try to build a prediction model by using random forest.

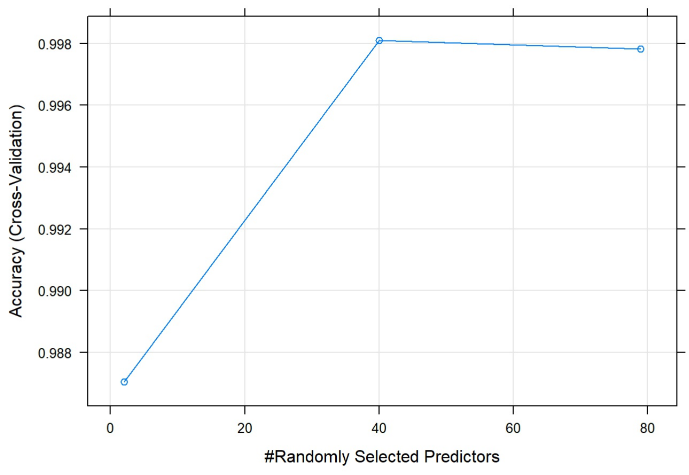
```
contrRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
RF <- train(classe ~ ., data=train, method="rf", trControl=contrRF)
RF$finalModel
```

```
## 
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 40
## 
##         OOB estimate of  error rate: 0.09%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3906    0    0    0    0 0.0000000000
## B    1 2657    0    0    0 0.0003762227
## C    0    3 2390    3    0 0.0025041736
## D    0    0    2 2248    2 0.0017761989
## E    0    0    0    2 2523 0.0007920792
```

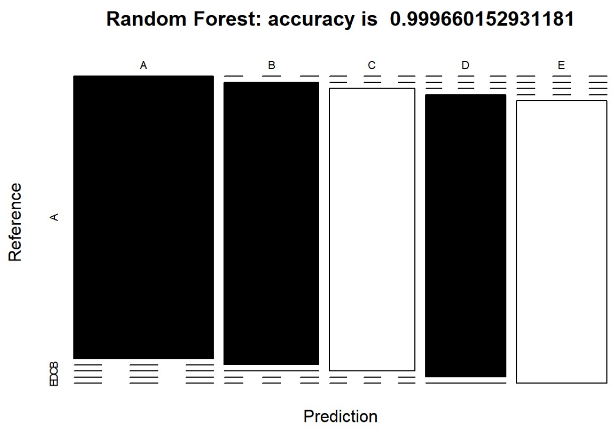
```
predict1 <- predict(RF, newdata=test)
forest <- confusionMatrix(predict1, test$classe)
forest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    1    0    0
##          C    0    0 1025    0    0
##          D    0    0    0  964    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9990   1.0000   0.9991
## Specificity            1.0000   0.9998   1.0000   0.9998   1.0000
## Pos Pred Value         1.0000   0.9991   1.0000   0.9990   1.0000
## Neg Pred Value         1.0000   1.0000   0.9998   1.0000   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1742   0.1638   0.1837
## Detection Prevalence   0.2845   0.1937   0.1742   0.1640   0.1837
## Balanced Accuracy      1.0000   0.9999   0.9995   0.9999   0.9995
```

```
plot(RF)
```



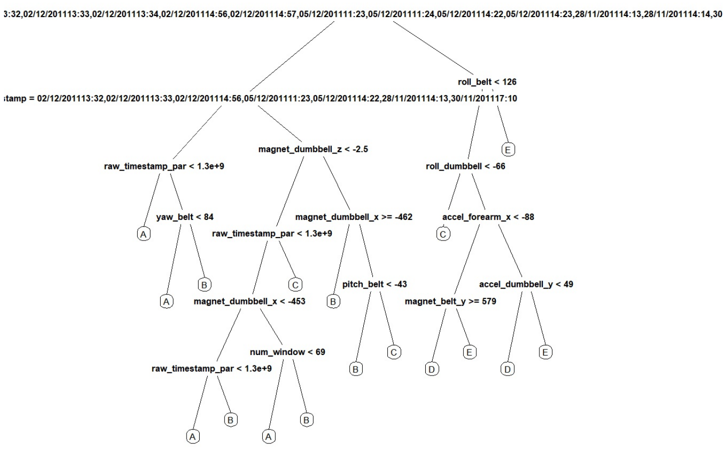
```
plot(forest$table, col = forest$byClass, main = paste("Random Forest: accuracy is ", forest$overall['Accuracy']
))
```



Accuracy is very high and it is 0.9997 about. It means that expected out-of-sample error is practically null.

Decision tree

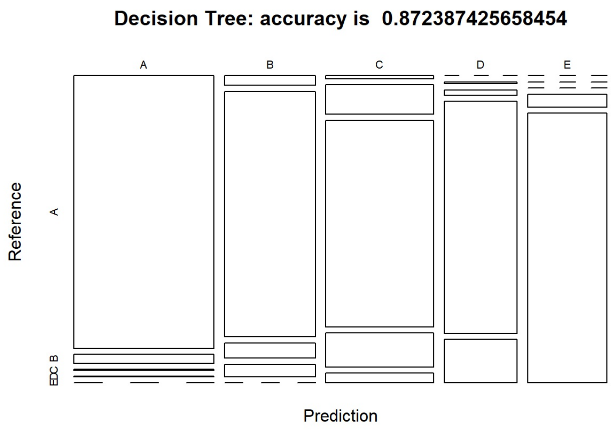
```
decisionTree <- rpart(classe ~ ., data=train, method="class")
prp(decisionTree)
```



```
predict2 <- predict(decisionTree, test, type = "class")
tree <- confusionMatrix(predict2, test$classe)
tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1624   54    4    2    0
##          B   37  943   59   47    0
##          C   13  136  947  156   44
##          D    0    6   16  716  134
##          E    0    0    0   43  904
##
## Overall Statistics
##
##                Accuracy : 0.8724
##                  95% CI : (0.8636, 0.8808)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8386
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9701   0.8279   0.9230   0.7427   0.8355
## Specificity            0.9858   0.9699   0.9282   0.9683   0.9910
## Pos Pred Value         0.9644   0.8683   0.7307   0.8211   0.9546
## Neg Pred Value         0.9881   0.9592   0.9828   0.9505   0.9640
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2760   0.1602   0.1609   0.1217   0.1536
## Detection Prevalence   0.2862   0.1845   0.2202   0.1482   0.1609
## Balanced Accuracy      0.9779   0.8989   0.9256   0.8555   0.9133
```

```
plot(tree$table, col = tree$byClass,
     main = paste("Decision Tree: accuracy is ", tree$overall['Accuracy']))
```



The accuracy is lower than the previous case. It is 0.87, so we have an expected error of 0.13 about.

Boosted regression

```
contrBR <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
BR<- train(classe ~ ., data=train, method = "gbm", trControl = contrBR, verbose = FALSE)
BR$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 79 predictors of which 47 had non-zero influence.
```

```
print(BR)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    57 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 12364, 12364, 12364, 12363, 12363, 12363, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8406477  0.7977560
##   1                  100      0.8991050  0.8722257
##   1                  150      0.9280774  0.9088971
##   2                   50      0.9561026  0.9444238
##   2                  100      0.9872600  0.9838854
##   2                  150      0.9920639  0.9899619
##   3                   50      0.9830374  0.9785389
##   3                  100      0.9938837  0.9922637
##   3                  150      0.9961408  0.9951186
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
predict3 <- predict(BR, newdata=test)
boosted <- confusionMatrix(predict3, test$classe)
boosted
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    1 1133    1    0    0
##          C    0    1 1015    0    0
##          D    0    2   10  960    5
##          E    0    0    0    4 1077
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9947   0.9893   0.9959   0.9954
## Specificity            0.9993   0.9996   0.9998   0.9965   0.9992
## Pos Pred Value         0.9982   0.9982   0.9990   0.9826   0.9963
## Neg Pred Value         0.9998   0.9987   0.9977   0.9992   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1925   0.1725   0.1631   0.1830
## Detection Prevalence   0.2848   0.1929   0.1726   0.1660   0.1837
## Balanced Accuracy      0.9993   0.9972   0.9945   0.9962   0.9973
```

By combining various predicting model, we now use a boosted regression model. This can be very useful to get further accuracy by combining weak predictors. However, accuracy is 0.9947 - predictors used were not so weak, indeed! We will then opt for random forest.

Predictions

Here you will find the predictions on the 20 cases from validation set.

```
Results <- predict(RF, newdata=validation)
Results
```

```
##  [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
```