Practical Machine Learning Course - Final Project

Muhammad Mahagne

January 28, 2016

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 (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 (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (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 (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.

Reproducibility

Load required libraries

```
library(caret)
library(ggplot2)
library(randomForest)
library(rpart)
library(rpart.plot)
library(caTools)
library(kernlab)
library(adabag)
library(plyr)

set.seed (2324)
```

Load the dataset

Set training and test data

```
trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

trainingFile <- "pm-training.csv"

if (file.exists(trainingFile)){
        training <- read.csv(trainingFile , na.strings = c("NA" , "") )
} else {
        download.file(trainingURL, trainingFile)
        training <- read.csv(trainingFile , na.strings = c("NA" , ""))
}

testingFile <- "pm-testing.csv"

if (file.exists(testingFile)){
        test <- read.csv(testingFile , na.strings = c("NA" , "") )
} else {
        download.file(testingURL, testingFile)
        test <- read.csv(testingFile , na.strings = c("NA" , ""))
}
```

Explore the training dataset

I'm not printing the results inside the report due to size

```
dim(training)
head(training)
summary(training)
```

Training and Test Sets - Cleaning and pre processing

All following steps will be run on both training and test data sets

Due to the large data set and in order to make it easier to work with it, I cleaned the dataset from the variables that have high share of NAs or variables which characterized by low variance

```
nearzero <- caret::nearZeroVar(training, saveMetrics = TRUE)
training <- training[, !nearzero$nzv]
test <- test[,!nearzero$nzv]
```

Remove variables with more than 50% missing values

```
varRemove <- sapply(colnames(training),
                    function(x) if(sum(is.na(training[, x])) > 0.50*nrow(training)){
                      return(TRUE)
                    }else{
                      return(FALSE)}
  )

training <- training[, !varRemove]
test <- test[, !varRemove]
```

Remove columns which dont impact the prediction (i.e. name , entry time and date etc.)

```
training <- training[,-(1:6)]
test <- test[,-(1:6)]
```

Find variables with high correlation (exclude “classe” variable) in order to remove variables based on pair-wise correlations.

```
high_cor <- caret::findCorrelation(cor(training [,-53]) , cutoff = 0.9)
names(training)[high_cor]
```

```
## [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
## [4] "accel_belt_x"     "gyros_dumbbell_x" "gyros_dumbbell_z"
## [7] "gyros_arm_x"
```

we can see that about 15% of the variables are correlated , therefore Principal component analysis (PCA) technique will be used.

Partioning the training set into two sets

```
inTrain <- caret::createDataPartition(y=training$classe, p=0.6, list=FALSE)
trainingDat <- training[inTrain, ]
testingDat <- training[-inTrain, ]
dim(trainingDat)
```

```
## [1] 11776    53
```

```
dim(testingDat)
```

```
## [1] 7846   53
```

Perform Cross Validation and Model Specification

I will use trainControl to run 5 fold cross valication with pca pre process option. It will help not to have overfitting in addition it will reduce number of predectors and reduce noise.

```
trainingControl <- caret::trainControl (method= "cv" ,number = 5 , preProcOptions = "pca"
,  verboseIter = FALSE , allowParallel = TRUE )
```

Now I will estimate two top performing algorithms ,Random Forest and boosting, then compare the accuracy to decide which model to use in order to predict classe in test data set.

Random Forest

```
rf <- caret::train(classe ~ ., data = trainingDat, method = "rf", trControl= trainingCont
rol)
pred_rf <- predict(rf,testingDat)
```

```
confusionMatrix(pred_rf, testingDat$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   12    0    0    0
##          B    2 1501    4    0    0
##          C    0    4 1360   13    4
##          D    0    1    4 1271    5
##          E    0    0    0    2 1433
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9935          
##                  95% CI : (0.9915, 0.9952)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9918          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9888   0.9942   0.9883   0.9938
## Specificity            0.9979   0.9991   0.9968   0.9985   0.9997
## Pos Pred Value         0.9946   0.9960   0.9848   0.9922   0.9986
## Neg Pred Value         0.9996   0.9973   0.9988   0.9977   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1913   0.1733   0.1620   0.1826
## Detection Prevalence   0.2858   0.1921   0.1760   0.1633   0.1829
## Balanced Accuracy      0.9985   0.9939   0.9955   0.9934   0.9967
```

Boosting

```
bs <- caret::train(classe ~ ., data = trainingDat, method = "gbm", trControl= trainingCon
trol)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.3
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
pred_bs <- predict(bs,testingDat)
```

```
confusionMatrix(pred_bs, testingDat$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2197   36    0    3    3
##          B   27 1444   42    5   13
##          C    3   34 1310   38   21
##          D    5    3   14 1232   22
##          E    0    1    2    8 1383
## 
## Overall Statistics
## 
##                Accuracy : 0.9643
##                  95% CI : (0.96, 0.9683)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9549
##  Mcnemar's Test P-Value : 8.395e-08
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9843   0.9513   0.9576   0.9580   0.9591
## Specificity            0.9925   0.9863   0.9852   0.9933   0.9983
## Pos Pred Value         0.9812   0.9432   0.9317   0.9655   0.9921
## Neg Pred Value         0.9938   0.9883   0.9910   0.9918   0.9909
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2800   0.1840   0.1670   0.1570   0.1763
## Detection Prevalence   0.2854   0.1951   0.1792   0.1626   0.1777
## Balanced Accuracy      0.9884   0.9688   0.9714   0.9757   0.9787
```

Conclusion

Comparing the results summary of the two models , I find that accurancy of Random Forest (~0.99) model is higher than boosting model (~0.96) . In addition the accurancy of Random Forest is more than 0.99 which is very high meaning model is very accurate.

Predict test data with random forest model in order to answer the assignment

```
test_pred_rf <- predict(rf,test)

n <- 20
for(i in 1:n){
                x <- paste("Answer",i, "=" ,test_pred_rf[i]   )
                print(x)
             }
```

```
## [1] "Answer 1 = B"
## [1] "Answer 2 = A"
## [1] "Answer 3 = B"
## [1] "Answer 4 = A"
## [1] "Answer 5 = A"
## [1] "Answer 6 = E"
## [1] "Answer 7 = D"
## [1] "Answer 8 = B"
## [1] "Answer 9 = A"
## [1] "Answer 10 = A"
## [1] "Answer 11 = B"
## [1] "Answer 12 = C"
## [1] "Answer 13 = B"
## [1] "Answer 14 = A"
## [1] "Answer 15 = E"
## [1] "Answer 16 = E"
## [1] "Answer 17 = A"
## [1] "Answer 18 = B"
## [1] "Answer 19 = B"
## [1] "Answer 20 = B"
```