Practical Machine Learning Peer Graded Assignment

Dhrubasattwata Roy Choudhury

12 June 2018

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.

The goal of this project is to predict the manner in which they did the exercise. This is the classe variable in the training set, considering any of the other variables as predictors.

How the model was built

The outcome variable is the classe, a factor variable of 5 levels:

- A - exactly according to the specification
- B - throwing the elbows to the front
- C - lifting the dumbbell only halfway
- D - lowering the dumbbell only halfway
- E - throwing the hips to the front

Class ***A*** corresponds to the specified execution of the exercise the 10 people have been asked, while the other classes correspond to common mistakes. All other variables in the data set will be used as predictor, after cleaning.
Models evaluation will be based on maximising the accuracy and minimazing the out of sample error. We will build two different models, using decision tree and random forests methods. The model with the highest accuracy will be choosen as the final Model and will be validated on the original testing data set.

Cross Validation

We will subset the training set into two subsamples:

- subtraing data set, as 70% of the original training data set
- testing data set, as 30% of the original training data set We will fit the model using the subtesting data set and test it on the testing data set. Once the best model is choosed, it will be validated on the original testing data set

Expected out of sample error

The expected out of sample error will correpond to the ***accuracy*** in the cross-validation data. Accuracy is the proportion between the correctly classified observation over the total observation in the subtraining set used for testing the model. Expected Accurracy is the accuracy in the out of sample data set, i.e. the

original testing data set. The ***expected out of sample error*** will then correspond to the proportion between the missclassified observation over the total observation in the original testing data set.

Necessary Packages and Set Seed

The following packages are required:

```
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(RColorBrewer
library(rattle)
library(ggplot2)
```

In order to ensure reproducibility of the analysis, we will set the seed to 1000

```
set.seed(1000)
```

Getting and Cleaning the data

Download the training and the testing data set from the provided web link:

```
##training data set
trainUrl <- "C:\\Users\\DHRUBASATTWATA\\Desktop\\Data Science John Hopkins\\pml-training.csv"
##testing data set
testUrl  <- "C:\\Users\\DHRUBASATTWATA\\Desktop\\Data Science John Hopkins\\pml-testing.csv"
##load the training data
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
##load the testing data
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

First, we split the training data set into two subsamples, the 70% will be used as training set and the remaining 30% will be used as testing set.

```
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
subTraining <- training[inTrain, ]
subTesting <- training[-inTrain, ]
```

```
dim(subTraining)
```

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

```
dim(subTesting)
```

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

We perform data cleaning in multiple steps, listed below.

We will remove the first 7 columns as they are metadata and not relevant for the prediction

```
subTraining <- subTraining[, -c(1:7)]
```

We will then clean the data set from all variables with high number of NAs and with variance close to 0.

```
##clean near zero variance variables
NearZeroVar <- nearZeroVar(subTraining, saveMetrics = TRUE)
subTraining <- subTraining[, NearZeroVar$nzv == FALSE]
```

Finally, we remove all variables that have more than 60% of NA values

```
subTraining_clean <- subTraining
for (i in 1:length(subTraining)){
  if(sum(is.na(subTraining[, i]))/ nrow(subTraining) >= .6) {
    for (j in 1:length(subTraining_clean)){
      if(length(grep(names(subTraining[i]), names(subTraining_clean)[j])) == 1){
        subTraining_clean <- subTraining_clean[, -j]
      }
    }
  }
}


subTraining <- subTraining_clean
```

Let's now apply the same transformation to the subTesting and Tesing data set

```
##clean the subtesting data set
clean1 <- colnames(subTraining)
```

```
subTesting <- subTesting[clean1]


##clean the testing data set (already with column Classe removed)
clean2 <- colnames(subTraining[, -53])
testing <- testing[clean2]
```

Model with Random Forests

We will build a random forests model on the subtraining data set and the validate it on the subtesting data set (30% of the original training data set).

All other variables are used to predict variable Classe.

```
##create the model with randomForest function
modFit_rf <- randomForest(classe ~., data = subTraining)
##calculate predicted values
pred_rf <- predict(modFit_rf, subTesting)
##calculate confusion matrix
cm_rf <- confusionMatrix(pred_rf, subTesting$classe)
print(cm_rf)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##         A 1671    5    0    0    0
##         B    3 1133    6    0    0
##         C    0    1 1018   10    0
##         D    0    0    2  953    3
##         E    0    0    0    1 1079
##
## Overall Statistics
##
##               Accuracy : 0.9947
##                 95% CI : (0.9925, 0.9964)
##    No Information Rate : 0.2845
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                  Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9982   0.9947   0.9922   0.9886   0.9972
## Specificity          0.9988   0.9981   0.9977   0.9990   0.9998
## Pos Pred Value       0.9970   0.9921   0.9893   0.9948   0.9991
## Neg Pred Value       0.9993   0.9987   0.9984   0.9978   0.9994
## Prevalence           0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate       0.2839   0.1925   0.1730   0.1619   0.1833
## Detection Prevalence   0.2848   0.1941   0.1749   0.1628   0.1835
## Balanced Accuracy      0.9985   0.9964   0.9950   0.9938   0.9985
```

Accuracy of the model is 99%, which means the expected out of sample error is 1%. Let's check the error rate for all the different trees evaluated by the model

```
plot(modFit_rf)
```

Error rate is always below 14% for all the 500 trees.

Model with Decision Tree

Let's now build a second model on the subtraining data set (60% of the original training data set). We will again use the remaing 30% of the training data set to test the model.

```
modFit_dt <- rpart(classe ~., data = subTraining, method = "class")

##calculate predicted values

pred_dt <- predict(modFit_dt, subTesting, type = "class")

##calculate the confusion matrix

cm_dt <- confusionMatrix(pred_dt, subTesting$classe)

print(cm_dt)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##         A 1461  150   10   57   50
##         B   88  814  138   68   71
##         C   14  111  790  161   83
##         D   44   56   47  643   82
##         E   67    8   41   35  796
##
## Overall Statistics
##
##               Accuracy : 0.7653
##                 95% CI : (0.7543, 0.7761)
##    No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##     Kappa : 0.7028
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.8728   0.7147   0.7700   0.6670   0.7357
## Specificity          0.9366   0.9231   0.9241   0.9535   0.9686
## Pos Pred Value       0.8455   0.6904   0.6816   0.7374   0.8405
## Neg Pred Value       0.9488   0.9309   0.9501   0.9360   0.9421
## Prevalence           0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate       0.2483   0.1383   0.1342   0.1093   0.1353
## Detection Prevalence 0.2936   0.2003   0.1969   0.1482   0.1609
## Balanced Accuracy    0.9047   0.8189   0.8470   0.8102   0.8521
```

Accuracy of the model is 68%, much lower compared to the first model built. The expected out of sample error is 32%.

Final Prediction

Considering the higher accuracy of the model built using the randon forests method, we will use the first model to predict on the testing data

```
finalPred <- predict(modFit_rf, testing, type = "class")

print(finalPred)
```

```
##  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

We have to compare two different machine learning algorithms which are well known for their ability of detecting the features that are important for classification: Random Forest and Decision Tree.

Based on the data we had, the Random Forest algorithm proved to be more accurate to predict the quality level of the exercise execution.