Practical Machine Learning Course Project

Mehul Singh

Saturday, February 18, 2017

(Background, Data and What you should submit sections are directly copied from course's assignment page)

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).

Data

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.

What you should submit

The goal of your 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.

Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Preliminary Work

Reproduceability

An overall pseudo-random number generator seed was set at 1234 for all code. In order to reproduce the results below, the same seed should be used.
Different packages were downloaded and installed, such as caret and randomForest. These should also be installed in order to reproduce the results below (please see code below for ways and syntax to do so).

How the model was built

Our outcome variable is classe, a factor variable with 5 levels. For this data set, “participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)

- throwing the elbows to the front (Class B)

- lifting the dumbbell only halfway (Class C)

- lowering the dumbbell only halfway (Class D)

- throwing the hips to the front (Class E)

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes." [1]
Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction.
Two models will be tested using decision tree and random forest algorithms. The model with the highest accuracy will be chosen as our final model.

Cross-validation

Cross-validation will be performed by subsampling our training data set randomly without replacement into 2 subsamples: subTraining data (75% of the original Training data set) and subTesting data (25%). Our models will be fitted on the subTraining data set, and tested on the subTesting data. Once the most accurate model is choosen, it will be tested on the original Testing data set.

Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Reasons for my choices

Our outcome variable “classe” is an unordered factor variable. Thus, we can choose our error type as 1-accuracy. We have a large sample size with N= 19622 in the Training data set. This allow us to divide our Training sample into subTraining and subTesting to allow cross-validation. Features with all missing values will be discarded as well as features that are irrelevant. All other features will be kept as relevant variables.

Decision tree and random forest algorithms are known for their ability of detecting the features that are important for classification [2]. Feature selection is inherent, so it is not so necessary at the data preparation phase. Thus, there won't be any feature selection section in this report.

Code and Results

Packages, Libraries, Seed

Installing packages, loading libraries, and setting the seed for reproducibility:

```
#install.packages("caret")

#install.packages("randomForest")

#install.packages("rpart")

library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest) #Random forest for classification and regression
```

```
## randomForest 4.6-10

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rpart) # Regressive Partitioning and Regression trees

library(rpart.plot) # Decision Tree plot



# setting the overall seed for reproducibility

set.seed(1234)
```

Loading data sets and preliminary cleaning

First we want to load the data sets into R and make sure that missing values are coded correctly. Irrelevant variables will be deleted.
Results will be hidden from the report for clarity and space considerations.

```
# After saving both data sets into my working directory

# Some missing values are coded as string "#DIV/0!" or "" or "NA" - these will be chan
ged to NA.
```

```
# We notice that both data sets contain columns with all missing values - these will b
e deleted.

# Loading the training data set into the  R session replacing all missing with "NA"

trainingset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv", na.strings=c("NA","#DIV/0!", ""))

# Loading the testing data set

testingset <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv', na.strings=c("NA","#DIV/0!", ""))

# Check dimensions for number of variables and number of observations

dim(trainingset)

dim(testingset)

# Delete columns with all missing values

trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]

testingset <-testingset[,colSums(is.na(testingset)) == 0]


# Some variables are irrelevant to our current project: user_name, raw_timestamp_part_
1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and  num_window (columns 1 to 7).
We can delete these variables.

trainingset   <-trainingset[,-c(1:7)]

testingset <-testingset[,-c(1:7)]


# and have a look at our new datasets:

dim(trainingset)

dim(testingset)

head(trainingset)

head(testingset)
```

Partitioning the training data set to allow cross-validation

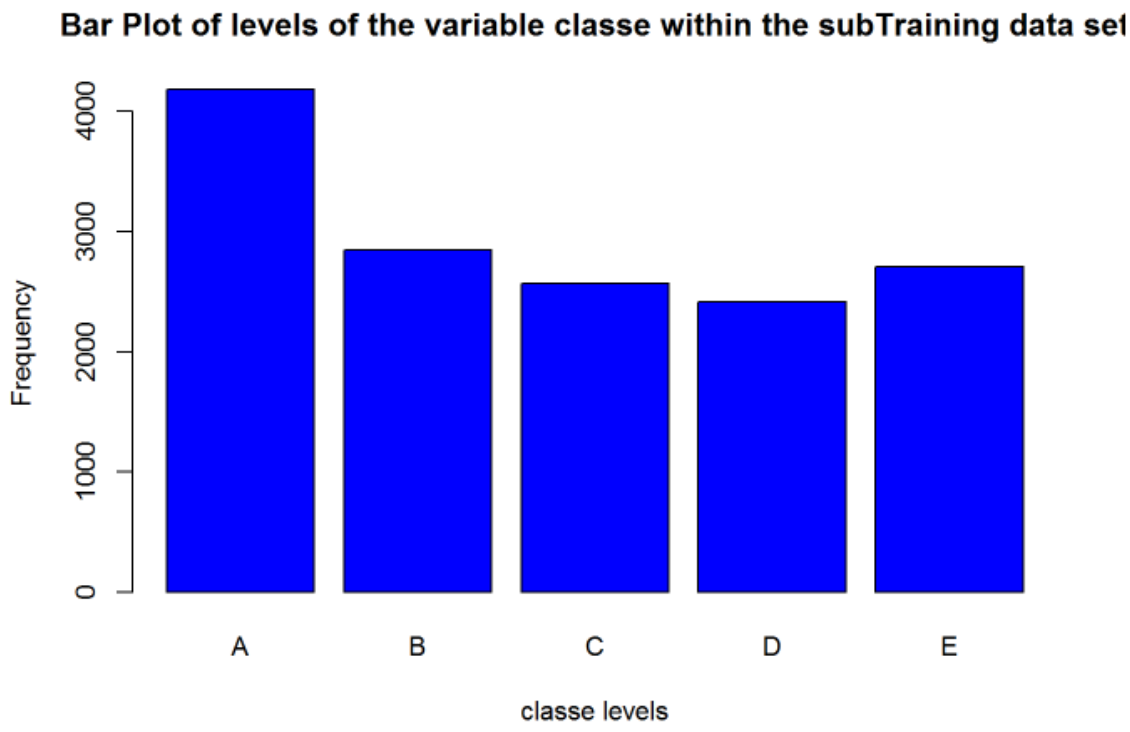
The training data set contains 53 variables and 19622 obs.
The testing data set contains 53 variables and 20 obs.
In order to perform cross-validation, the training data set is partionned into 2 sets: subTraining (75%) and subTest (25%).
This will be performed using random subsampling without replacement.

```
subsamples <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)

subTraining <- trainingset[subsamples, ]

subTesting <- trainingset[-subsamples, ]

dim(subTraining)

dim(subTesting)

head(subTraining)

head(subTesting)
```

A look at the Data

The variable “classe” contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the subTraining data set and compare one another.

```
plot(subTraining$classe, col="blue", main="Bar Plot of levels of the variable classe within the subTraining data set", xlab="classe levels", ylab="Frequency")
```

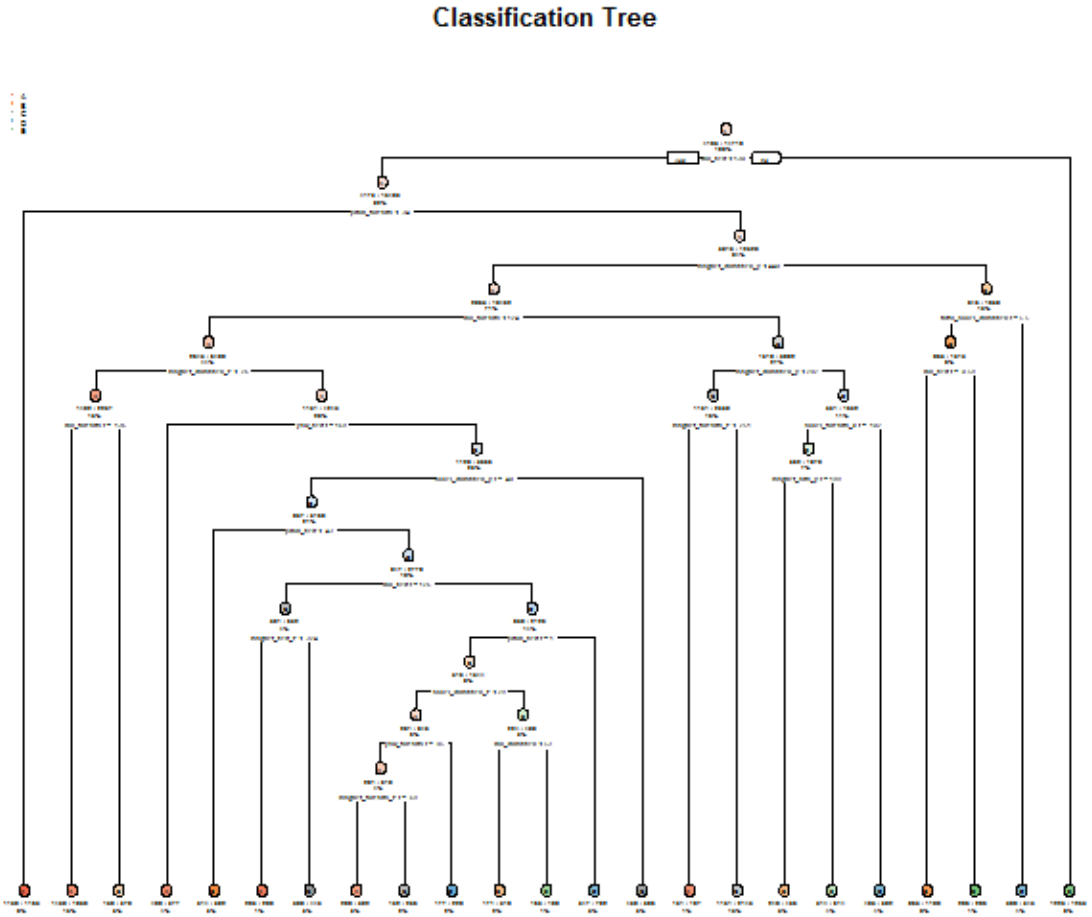


From the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent with more than 4000 occurrences while level D is the least frequent with about 2500 occurrences.

First prediction model: Using Decision Tree

```
model1 <- rpart(classe ~ ., data=subTraining, method="class")
```

```r
# Predicting:
prediction1 <- predict(model1, subTesting, type = "class")

# Plot of the Decision Tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```




```r
# Test results on our subTesting data set:
confusionMatrix(prediction1, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1235  157   16   50   20
```

```
##          B   55  568   73   80  102
##          C   44  125  690  118  116
##          D   41   64   50  508   38
##          E   20   35   26   48  625
##
## Overall Statistics
##
##                Accuracy : 0.739
##                  95% CI : (0.727, 0.752)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.67
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.885    0.599    0.807    0.632    0.694
## Specificity             0.931    0.922    0.900    0.953    0.968
## Pos Pred Value          0.836    0.647    0.631    0.725    0.829
## Neg Pred Value          0.953    0.905    0.957    0.930    0.933
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.252    0.116    0.141    0.104    0.127
## Detection Prevalence    0.301    0.179    0.223    0.143    0.154
## Balanced Accuracy       0.908    0.760    0.854    0.792    0.831
```

Second prediction model: Using Random Forest

```
model2 <- randomForest(classe ~. , data=subTraining, method="class")
# Predicting:
prediction2 <- predict(model2, subTesting, type = "class")
# Test results on subTesting data set:
confusionMatrix(prediction2, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    3    0    0    0
##          B    1  944   10    0    0
##          C    0    2  843    6    0
##          D    0    0    2  798    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.995
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.284
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.995    0.986    0.993    1.000
## Specificity             0.999    0.997    0.998    1.000    1.000
## Pos Pred Value          0.998    0.988    0.991    0.997    1.000
## Neg Pred Value          1.000    0.999    0.997    0.999    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.172    0.163    0.184
## Detection Prevalence    0.285    0.195    0.174    0.163    0.184
## Balanced Accuracy       0.999    0.996    0.992    0.996    1.000
```

Decision

As expected, Random Forest algorithm performed better than Decision Trees.
Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. **The random Forest model is chosen**. The accuracy of the model is 0.995. The expected out-of-sample error is estimated at 0.005, or **0.5%**. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be misclassified.

Submission

```
# predict outcome levels on the original Testing data set using Random Forest algorith
m

predictfinal <- predict(model2, testingset, type="class")
predictfinal
```

```
##  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
```

```
# Write files for submission
pml_write_files = function(x){
n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(predictfinal)
```