Practical Machine Learning: Course Project

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.

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.

1. 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 :-).
2. 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.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Building the model

```
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

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

#library(RColorBrewer)
#library(rattle)

 # Reading the training data -replacing all missing with "NA"
trainingset <- read.csv("C:/machinelearning/pml-training.csv", na.strings=c("NA","#D
IV/0!", ""))

# Reading the testing data set
testingset <- read.csv('C:/machinelearning/pml-testing.csv', na.strings=c("NA","#DI
V/0!", ""))

# Check dimensions for number of variables and number of observations
dim(trainingset)
```

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

```
dim(testingset)
```

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

```
# Cleaning data
# Delete the columns with missing values
trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]

# get rid of the variables we do not need: user_name, raw_timestamp_part_1, raw_time
stamp_part_,2 cvtd_timestamp, new_window, and  num_window (columns 1 to 7). We can d
elete these variables.
trainingset   <-trainingset[,-c(1:7)]
testingset <-testingset[,-c(1:7)]
dim(trainingset)
```

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

```
dim(testingset)
```

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

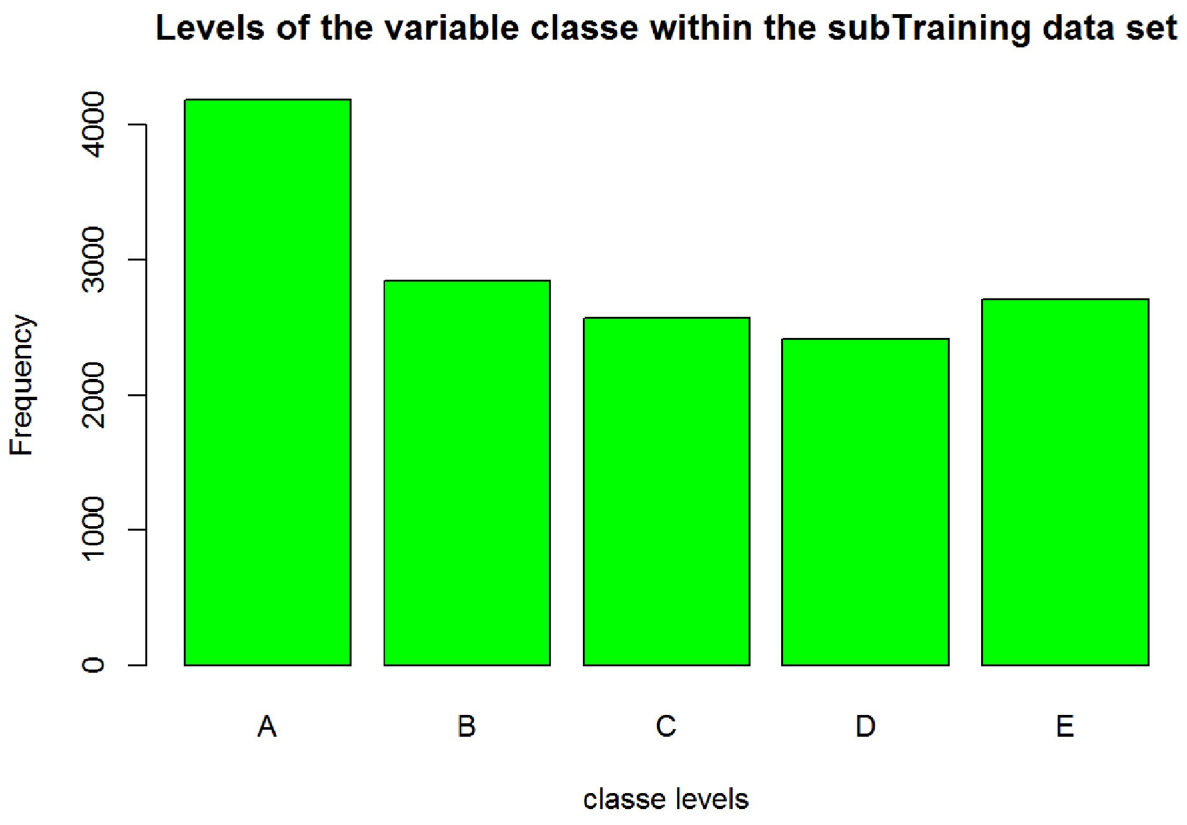
```
#head(trainingset)
#head(testingset)

#Partitioning the training data setfor cross-validation
#The training data set contains 53 variables and 19622 obs.
#The testing data set contains 53 variables and 20 obs.
#For cross-validation, the training data set is partionned into 2 sets: subTraining
(75%) and subTest (25%). This will be performed using random subsampling without rep
lacement.
subsamples <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
subTraining <- trainingset[subsamples, ]
subTesting <- trainingset[-subsamples, ]
dim(subTraining)
```

```
## [1] 14718    53
```

```
dim(subTesting)
```

```
## [1] 4904   53
```

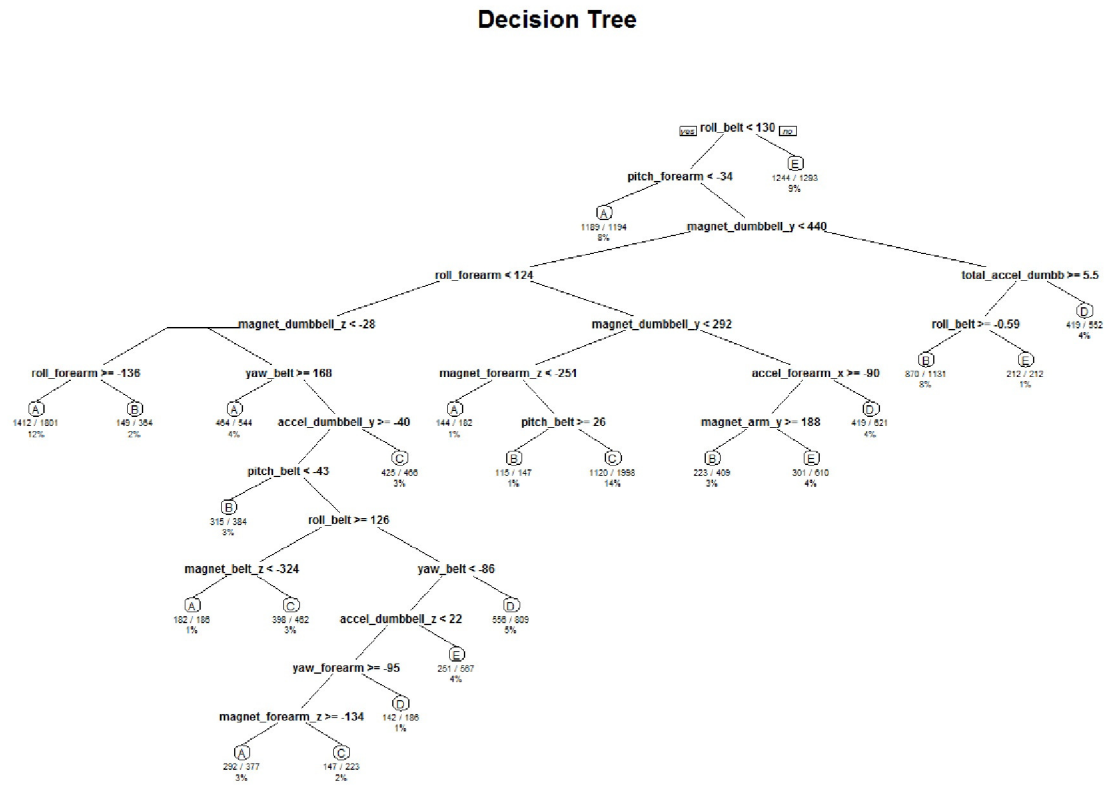
```
#head(subTraining)
#head(subTesting)
# Data Exploration
# “classe” has 5 levels: A, B, C, D and E. We plot  the outcome variable  to see th
e frequenc.
plot(subTraining$classe, col="green", main="Levels of the variable classe within th
e subTraining data set", xlab="classe levels", ylab="Frequency")
```



```
# The shows that Level A is the most frequent with more than 4000 occurrences while
level D is the least frequent with about 2500 occurrences.
```

Cross Validation

```
# Prediction model 1:  Decision Tree

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

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

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



```
# Test results -confusion matrix 1:
confusionMatrix(prediction1, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1231  153   15   43   11
##          B   43  561   77   68   65
##          C   30   87  694  121   96
##          D   50   59   49  487   44
##          E   41   89   20   85  685
##
## Overall Statistics
##
##                Accuracy : 0.7459
##                  95% CI : (0.7335, 0.7581)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6782
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8824   0.5911   0.8117  0.60572   0.7603
## Specificity            0.9367   0.9360   0.9175  0.95073   0.9413
## Pos Pred Value         0.8472   0.6892   0.6751  0.70682   0.7446
## Neg Pred Value         0.9525   0.9051   0.9585  0.92479   0.9458
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
## Detection Rate         0.2510   0.1144   0.1415  0.09931   0.1397
## Detection Prevalence   0.2963   0.1660   0.2096  0.14050   0.1876
## Balanced Accuracy      0.9096   0.7636   0.8646  0.77823   0.8508
```

```
#Prediction model 2: Random Forest
model2 <- randomForest(classe ~. , data=subTraining, method="class")

# Predicting:
prediction2 <- predict(model2, subTesting, type = "class")

# Test results -confusion matrix 2:
confusionMatrix(prediction2, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  945    1    0    0
##          C    0    0  853    6    1
##          D    0    0    1  797    1
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.995, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9958   0.9977   0.9913   0.9978
## Specificity            0.9989   0.9997   0.9983   0.9995   0.9998
## Pos Pred Value         0.9971   0.9989   0.9919   0.9975   0.9989
## Neg Pred Value         1.0000   0.9990   0.9995   0.9983   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1927   0.1739   0.1625   0.1833
## Detection Prevalence   0.2853   0.1929   0.1754   0.1629   0.1835
## Balanced Accuracy      0.9994   0.9978   0.9980   0.9954   0.9988
```

Model selection and expected out of sample error.

Model Selection

The Random Forest model performed better than the Decision Trees model. Accuracy of the Random Forest model is 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 selected. 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 should expect that very few of the test samples may be missclassified.

Final prediction and file creation for submission

```
# predict outcome levels with the original Testing data with the Random Forest model
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)
```