Practical Machine Learning Peer assignment

Sanya Sharma

August 4, 2018

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.

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].

Packages, Libraries and Seed

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

```r
library(caret)
```

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

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

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## 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.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
set.seed(1234)
```

Getting and cleaning data

The training data set can be found on the following URL:

```r
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

The testing data set can be found on the following URL:

```r
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Load Data

```r
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Partioning the training set into two

Partioning Training data set into two data sets, 60% for myTraining, 40% for myTesting:

```r
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Cleaning the data

The following transformations were used to clean the data:

Transformation 1: Cleaning NearZeroVariance Variables Run this code to view possible NZV Variables:

```r
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
```

Another subset of NZV variables.

```
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]

dim(myTraining)
```

```
## [1] 11776   100
```

Transformation 2: Killing first column of Dataset - ID Removing first ID variable so that it does not interfer with ML Algorithms:

```
myTraining <- myTraining[c(-1)]
```

Transformation 3: Cleaning Variables with too many NAs. For Variables that have more than a 60% threshold of NA's I'm going to leave them out:

```
trainingV3 <- myTraining #creating another subset to iterate in loop
for(i in 1:length(myTraining)) { #for every column in the training dataset
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { #if n?? NAs > 60% of total observati
ons
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  { #if the columns are the s
ame:
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}
#To check the new N?? of observations
dim(trainingV3)
```

```
## [1] 11776    58
```

```
#Setting back to our set:
myTraining <- trainingV3
rm(trainingV3)
```

Now let us do the exact same 3 transformations for myTesting and testing data sets.

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]

#To check the new N?? of observations
dim(myTesting)
```

```
## [1] 7846   58
```

```
#To check the new N?? of observations
dim(testing)
```

```
## [1] 20 57
```

In order to ensure proper functioning of Decision Trees and especially RandomForest Algorithm with the Test data set (data set provided), we need to coerce the data into the same type.

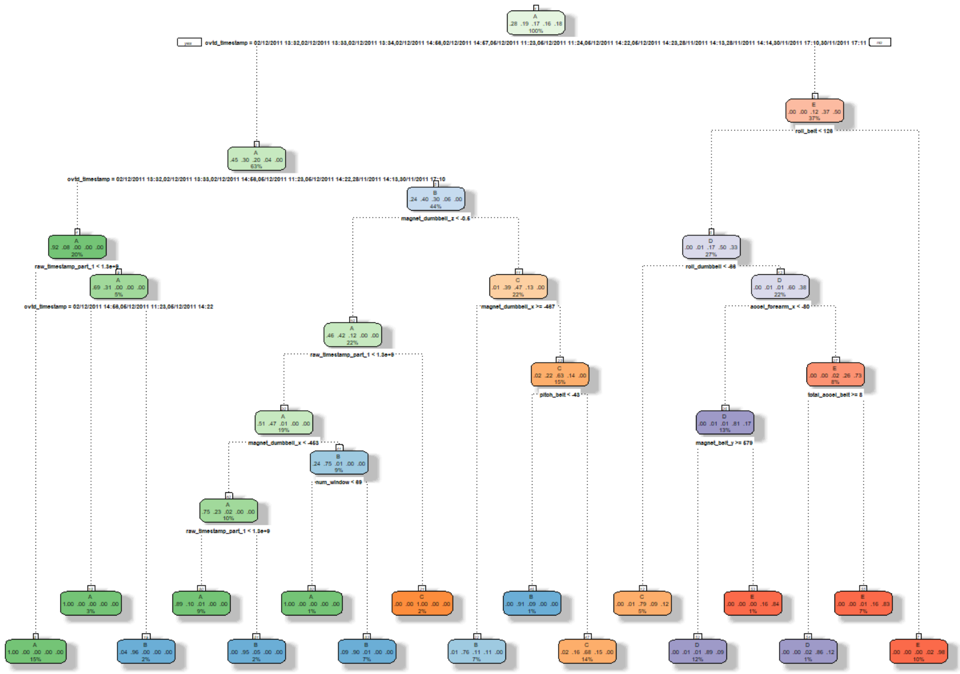
```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
#And to make sure Coertion really worked, simple smart ass technique:
testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not mean anything, this will be removed righ
t.. now:
testing <- testing[-1,]
```

Using ML algorithms for prediction: Decision Tree

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
```

To view the decision tree with fancy :

```
fancyRpartPlot(modFitA1)
```



Rattle 2018-Aug-04 23:35:50 sanya

Predicting:

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
```

```
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2161   61    5    3    0
##          B   50 1271   95   64    0
##          C   21  177 1242  203   65
##          D    0    9   19  899   92
##          E    0    0    7  117 1285
## 
## Overall Statistics
##                                           
##                Accuracy : 0.8741          
##                  95% CI : (0.8665, 0.8813)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.8407          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9682   0.8373   0.9079   0.6991   0.8911
## Specificity            0.9877   0.9670   0.9281   0.9817   0.9806
## Pos Pred Value         0.9691   0.8588   0.7272   0.8822   0.9120
## Neg Pred Value         0.9874   0.9612   0.9795   0.9433   0.9756
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2754   0.1620   0.1583   0.1146   0.1638
## Detection Prevalence   0.2842   0.1886   0.2177   0.1299   0.1796
## Balanced Accuracy      0.9779   0.9021   0.9180   0.8404   0.9359
```

Using ML algorithms for prediction: Random Forests

```
modFitB1 <- randomForest(classe ~. , data=myTraining)
```

Predicting in-sample error:

```
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    3    0    0
##          C    0    0 1363    5    0
##          D    0    0    2 1280    1
##          E    0    0    0    1 1441
## 
## Overall Statistics
## 
##                Accuracy : 0.9982
##                  95% CI : (0.997, 0.999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9977
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9963   0.9953   0.9993
## Specificity            0.9996   0.9995   0.9992   0.9995   0.9998
## Pos Pred Value         0.9991   0.9980   0.9963   0.9977   0.9993
## Neg Pred Value         1.0000   0.9997   0.9992   0.9991   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1737   0.1631   0.1837
## Detection Prevalence   0.2847   0.1936   0.1744   0.1635   0.1838
## Balanced Accuracy      0.9998   0.9991   0.9978   0.9974   0.9996
```

Generating Files to submit as answers for the Assignment:

Finally, using the provided Test Set out-of-sample error.

For Random Forests we use the following formula, which yielded a much better prediction in in-sample:

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
```

Function to generate files with predictions to submit for assignment

```
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(predictionsB2)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.