Personal Activity Prediction

Omyung Richard Kwon

July 16, 2017

Introduction

The goal of this project is to predict the manner in which test subjects did the exercise. This is the "classe" variable in the training set. I may use any of the other variables to predict with. I am to create a report describing how I built my model, how I used cross validation, what I think the expected out of sample error is, and why I made the choices you did. Finally, I will also use the prediction model to predict 20 different test cases

Following deliverables will be submitted * A link to the Github repo * repo will contain R markdown file * repo will contain a compiled HTML file w/ all the result

Load & Prepare Data

Both training and test data files are pre-downloaded to the local folder.

```
#rm(list=ls())

# set working directory where the data files are located
setwd("C:\\Users\\kwonr\\Projects\\Machine Learning")

raw.train <- read.csv("pml-training.csv", na.strings="NA")
raw.test <- read.csv("pml-testing.csv", na.strings="NA")

dim (raw.train)
```

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

```
dim (raw.test)
```

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

```
set.seed(1534)
```

Trim the unnecessary predictors. The original data contained 159 predictor variables. After getting rid of mostly NULL columns and near zero columns, I am left with 53 predictors + 1 category column. All the credit for this process goes to the folks who have posted this topic on the Week 4 forum.

The training data is split into 60/40 segments so that we can estimate the out-of-sample error.

```
library(caret)
raw.train <- raw.train[, colSums(is.na(raw.train)) < nrow(raw.train) * 0.95]
nearzero <- nearZeroVar(raw.train)
raw.train <- raw.train[, -nearzero]
raw.train <- subset(raw.train,
                    select=-c(X,user_name,raw_timestamp_part_1,raw_timestamp_pa
rt_2,cvtd_timestamp))

#60/40 split
inTrain <- createDataPartition(y=raw.train$classe, p=0.6, list=FALSE)
training <- raw.train[inTrain,]
testing <- raw.train[-inTrain,]
```

Modeling & Prediction

This is a classification problem. So, I want to model using two popular and highly accurate modeling techniques - random forest and boosting.

The random forest modeling took a long time initially. Too long. So, going back to the Week 4 forum, I see that other people had same problem and also saw Len's suggestion on how to improve the performance. Read about it at below location. [Len's Suggestion on improving the performance of Random Forest model] (https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md (https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md))

Random Forest modeling first via caret package. 10-fold cross validation used

```
#random forest

library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
set.seed(12345)

cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

fitControl <- trainControl(method="cv", number=10, allowParallel=TRUE)

model_rf <- train(y=training$classe,x=training[,-54],
                  method="rf", data=training,
                  trControl=fitControl)

stopCluster(cluster)
registerDoSEQ()

#model_rf$finalModel

pred_rf <- predict(model_rf, newdata=testing)
confusionMatrix(pred_rf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    5    0    0    0
##          B    0 1508    8    0    0
##          C    0    5 1360    1    0
##          D    0    0    0 1285    2
##          E    0    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9959, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9942   0.9992   0.9986
## Specificity            0.9991   0.9987   0.9991   0.9997   1.0000
## Pos Pred Value         0.9978   0.9947   0.9956   0.9984   1.0000
## Neg Pred Value         1.0000   0.9984   0.9988   0.9998   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1733   0.1638   0.1835
## Detection Prevalence   0.2851   0.1932   0.1741   0.1640   0.1835
## Balanced Accuracy      0.9996   0.9961   0.9966   0.9995   0.9993
```

Random Forest Model:
* Accuracy: 0.9977 (99.8%)
* Out of Sample Error: 1 - 0.9977 = .0023 (0.23%)

Now, onto the boosting modeling using gbm (boosting with trees). Same 10-fold cross validation was used

```
set.seed(54321)

fitControl <- trainControl(method="cv",number=10)
model_gbm <- train(classe ~ ., data=training, method="gbm",
                   trControl=fitControl, verbose=FALSE)
```

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

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

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

```
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

```
#model_gbm$finalModel

pred_gbm <- predict(model_gbm, newdata=testing)
confusionMatrix(pred_gbm, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225   22    0    0    0
##          B    2 1484   15    7    5
##          C    0   11 1351    6    0
##          D    3    1    2 1272   15
##          E    2    0    0    1 1422
##
## Overall Statistics
##
##                Accuracy : 0.9883
##                  95% CI : (0.9856, 0.9905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9852
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9969   0.9776   0.9876   0.9891   0.9861
## Specificity            0.9961   0.9954   0.9974   0.9968   0.9995
## Pos Pred Value         0.9902   0.9808   0.9876   0.9838   0.9979
## Neg Pred Value         0.9987   0.9946   0.9974   0.9979   0.9969
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2836   0.1891   0.1722   0.1621   0.1812
## Detection Prevalence   0.2864   0.1928   0.1744   0.1648   0.1816
## Balanced Accuracy      0.9965   0.9865   0.9925   0.9930   0.9928
```

GBM Boosting Model:
* Accuracy: 0.9883 (98.8%)
* Out of Sample Error: 1 - 0.9883 (1.17%)

The Random Forest result shows the better and more accurate result.

Final Test Set Prediction

Using the random forest model, run the prediction against the final test data set.

```
all_predictors <- colnames(training)

new.test <- raw.test[, names(raw.test) %in% all_predictors]
predFinal <- predict (model_rf, newdata=new.test)
predFinal
```

```
##  [1] 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
```