Practical Machine Learning Project

By Amlish

Date 07/30/2016

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).

Some R packages

```
## Warning: package 'knitr' was built under R version 3.2.5
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
## Warning: package 'rattle' was built under R version 3.2.5
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## randomForest 4.6-12
```

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

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

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

```
## Warning: package 'corrplot' was built under R version 3.2.5
```

Downloading and Cleaning the Data

```
training<-read.csv("pml-training.csv", header=TRUE)
testing<-read.csv("pml-testing.csv", header=TRUE)
```

Data partitionn into two sets

```
set.seed(777)
intrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
train1<-training[intrain,]
train2<- training[-intrain,]
dim(train1)
```

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

```
dim(train2)
```

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

The training data set is divided into two datasets, train1 data set and train2 dataset. Both data subsets have 160 columns, that is number of variables.In the validation data set only the non zero values are needed.Thus we have to remove the NA.The near zero variance are also removed.

Removing near zero variance

```
NZV <- nearZeroVar(train1)
train1<- train1[,-NZV]
train2<-train2[,-NZV]
dim(train1)
```

```
## [1] 13737   106
```

```
dim(train2)
```

```
## [1] 5885  106
```

Removing all NAs and almost NAs

```
set.seed(777)
AllNA <- sapply(train1, function(x) mean(is.na(x))) > 0.95
train1 <- train1[, AllNA==F]
train2 <- train2[, AllNA==F]
dim(train1)
```

```
## [1] 13737    59
```

```
dim(train2)
```

```
## [1] 5885   59
```

Removing variables that do not make intuitive sense for prediction

```
set.seed(777)
train1 <- train1[, -(1:5)]
train2 <- train2[, -(1:5)]
dim(train1)
```

```
## [1] 13737    54
```

```
dim(train2)
```

```
## [1] 5885   54
```

Model Building and Prediction with Random Forest

```
set.seed(777)
library(randomForest)
library(ggplot2)
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=train1, method="rf", trControl=fitControl)
fit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B   10 2644    4    0    0 0.0052671181
## C    0    4 2392    0    0 0.0016694491
## D    0    0    8 2244    0 0.0035523979
## E    0    1    0    4 2520 0.0019801980
```

```
set.seed(123)
modFit1 <- randomForest(classe ~ ., data=train1)
prediction1 <- predict(modFit1, train2, type = "class")
cmrf <- confusionMatrix(prediction1, train2$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1136    2    0    0
##          C    0    0 1024    4    0
##          D    0    0    0  959    4
##          E    0    0    0    1 1078
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9981   0.9948   0.9963
## Specificity            0.9993   0.9996   0.9992   0.9992   0.9998
## Pos Pred Value         0.9982   0.9982   0.9961   0.9958   0.9991
## Neg Pred Value         1.0000   0.9994   0.9996   0.9990   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1740   0.1630   0.1832
## Detection Prevalence   0.2850   0.1934   0.1747   0.1636   0.1833
## Balanced Accuracy      0.9996   0.9985   0.9986   0.9970   0.9980
```

Prediction with Decision Trees

```
set.seed(123)
modFit2 <- rpart(classe ~ ., data=train1, method="class")
prediction2 <- predict(modFit2, train2, type = "class")
cmtree <- confusionMatrix(prediction2, train2$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521  176   49   89   28
##          B   44  680   58   25   17
##          C   44  110  783   39   14
##          D   58  171  136  754  166
##          E    7    2    0   57  857
##
## Overall Statistics
##
##                Accuracy : 0.7808
##                  95% CI : (0.77, 0.7913)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7222
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086   0.5970   0.7632   0.7822   0.7921
## Specificity            0.9188   0.9697   0.9574   0.8921   0.9863
## Pos Pred Value         0.8164   0.8252   0.7909   0.5868   0.9285
## Neg Pred Value         0.9620   0.9093   0.9504   0.9543   0.9547
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2585   0.1155   0.1331   0.1281   0.1456
## Detection Prevalence   0.3166   0.1400   0.1682   0.2184   0.1568
## Balanced Accuracy      0.9137   0.7833   0.8603   0.8371   0.8892
```

Prediction with Generalized Boosted Regression

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.2.5
```

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

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

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

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

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
set.seed(12345)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

GbmFit1 <- train(classe ~ ., data=train1, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)
```

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

```
## Warning: package 'plyr' was built under R version 3.2.5
```

```
GbmFinMod1 <- GbmFit1$finalModel

GbmPredTest <- predict(GbmFit1, newdata=train2)
GbmAccuracyTest <- confusionMatrix(GbmPredTest, train2$classe)
GbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    9    0    0    1
##          B    1 1114   11    8    5
##          C    0   13 1013    8    4
##          D    0    3    2  946    6
##          E    0    0    0    2 1066
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9844, 0.9903)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9781   0.9873   0.9813   0.9852
## Specificity            0.9976   0.9947   0.9949   0.9978   0.9996
## Pos Pred Value         0.9941   0.9781   0.9759   0.9885   0.9981
## Neg Pred Value         0.9998   0.9947   0.9973   0.9963   0.9967
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1893   0.1721   0.1607   0.1811
## Detection Prevalence   0.2860   0.1935   0.1764   0.1626   0.1815
## Balanced Accuracy      0.9985   0.9864   0.9911   0.9895   0.9924
```

From the three algorithms the random forest has the greatest accuracy which is 99.8%.That means,the random forset model is selected from the other two.Thus,the model fit on train is used to predict the label for the observations in testing data set.Then those predictions are writen to individual files.

Predicting Results on the Testing Data set

```
prediction3<- predict(modFit1, testing, type = "class")
prediction3
```

```
##  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
```

Writing the results to a text file 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=T)
    }
}
pml_write_files(prediction3)
```