Practical Machine Learning - Prediction Assignment Writeup

by: John Bohorquez

Summary

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The purpose of this project is to predict the manner in which they did the exercise.

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.

1: Load the data

The information is available from the website http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. Section Weight Lifting Exercise Dataset.

Datasets

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

Loading data

```
train_website <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(train_website, destfile = "./train.csv")
train_data <- read.table("./train.csv", sep =",", header = T)
test_website <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(test_website, destfile = "./test.csv")
test_data <- read.table("./test.csv", sep =",", header = T)
```

2: Explore and clean the data

Perform a general overview of the datasets.

```
## Check basic information to determine the dataset size.
library(caret)
```

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

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

```
dim(train_data)
dim(test_data)
colnames(train_data)
head(train_data[1:6, 1:10])
##  Clean the datasets with nearZeroVar to excluded worthlerss variables.
```

```
zero_var <- nearZeroVar(train_data)
train_data1 <- train_data[, -zero_var]
test_data1 <- test_data[, -zero_var]
dim(train_data1)
## Clean the datasets by excluding NA Columns (Threshold 80%).
na_column <- sapply(train_data1, function(x) mean(is.na(x))) > 0.8
train_data2 <- train_data1[, na_column == FALSE]
test_data2 <- test_data1[, na_column == FALSE]
dim(train_data2)
## Removing labeled columns that are not part of the records.
train_data3 <- train_data2[, 7:59]
test_data3 <- test_data2[, 7:59]
dim(train_data3)
```

3: Data Subseting (Training & testing)

Set a training partition in about 60% and 40% for testing purposes.

```
train_split <- createDataPartition(train_data3$classe, p=0.6, list=FALSE)
train_sample <- train_data3[train_split,]
test_sample <- train_data3[-train_split,]
dim(train_sample)
dim(test_sample)
```

4: Prediction Models

Analyze different models to see which one has the higher accuracy

4.1: Random Forest

All Variables

```
## This option contains all the variables
modelrfcv <- train(classe ~ ., data=train_sample, ntree=100, method='rf', trControl=trainControl(method=
```

```
set.seed(12345)
rfcv_predict <- predict(modelrfcv, test_sample)
rfcv_predict_conf <- confusionMatrix(rfcv_predict, test_sample$classe)
```

PCA preProcessing

```
## PCA method might shrink the predictors
modelrfcvpca <- train(classe ~ ., data=train_sample, ntree=100, method='rf', preProcess="pca", trContro
modelrfcvpca$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.87%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3294   20   20   10    4  0.01612903
```

```
## B   54 2155   57    3   10  0.05440983
## C   13   44 1963   28    6  0.04430380
## D    7    9  100 1808    6  0.06321244
## E    4   18   27   16 2100  0.03002309
```

```
set.seed(12345)
rfcvpca_predict <- predict(modelrfcvpca, test_sample)
rfcvpca_predict_conf <- confusionMatrix(rfcvpca_predict, test_sample$classe)
```

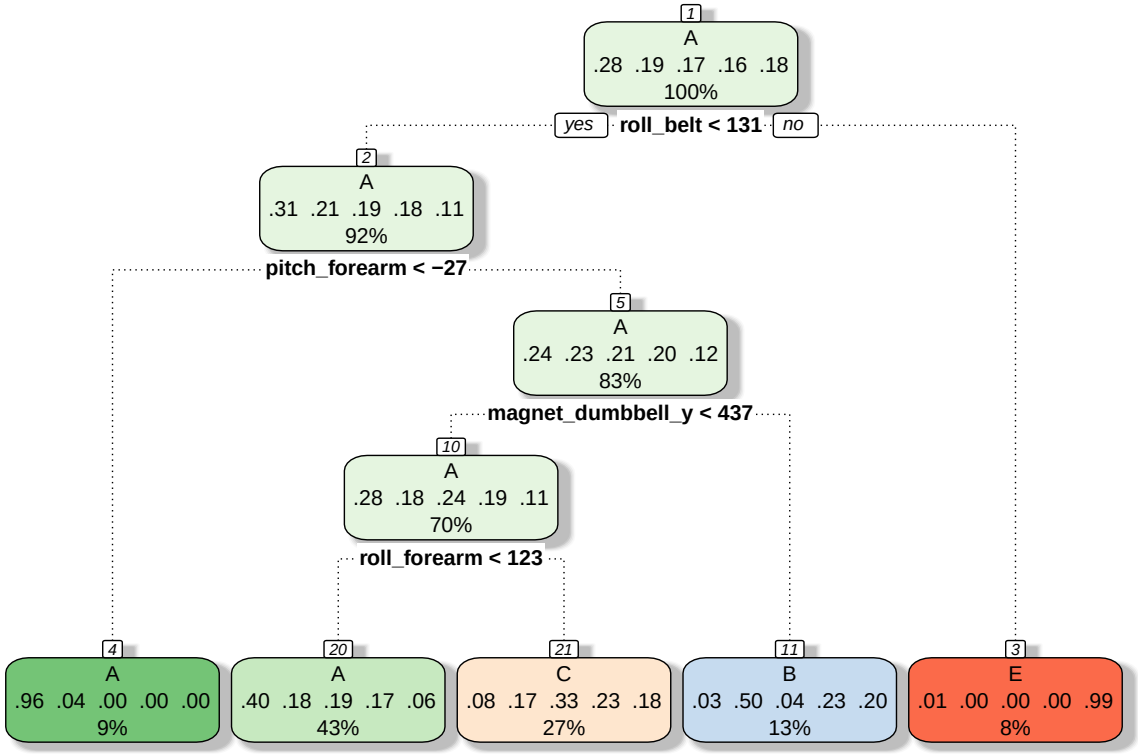
4.2: Decision Tree

Les't run this model with prediction portion

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
modeldt <- train(classe ~. , data=train_sample, method= "rpart")
fancyRpartPlot(modeldt$finalModel)
```




```
set.seed(12345)
dt_predict <- predict(modeldt, test_sample)
dt_predict_conf <- confusionMatrix(dt_predict, test_sample$classe)
```

4.3: Boosting (GBM)

Last model to do a comparison with early options

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(12345)
modelgbm <- train(classe~., data=train_sample, method="gbm", verbose= FALSE)
```

```
gbm_predict <- predict(modelgbm, test_sample)
gbm_predict_conf <- confusionMatrix(gbm_predict, test_sample$classe)
```

Comparison to select the fittest model

```
rfcv_predict_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9905684      0.9880665      0.9881738      0.9925871      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
rfcvpca_predict_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9681366      0.9596753      0.9640085      0.9719120      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
dt_predict_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.4951568      0.3395301      0.4840336      0.5062836      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
gbm_predict_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.625287e-01   9.525909e-01   9.580873e-01   9.666220e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   8.683668e-08
```

5: Conclusion

Based on the statistics the best model for our project is Random Forest (Without PCA preProcessing), since the accurracy is higher than other options.

Cross Validation

This tool is useful with procesess or databases that do not have a wide variation along the time, as we saw in the Forecast slides. For personal purposes, I'm interested to evaluate seasonal behaviors to enhance the prediction process in my field.

```
rfcv_predict_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 2231   19    0    1    0
##          B    1 1494    8    0    0
##          C    0    5 1355   24    3
##          D    0    0    5 1257    4
##          E    0    0    0    4 1435
##
## Overall Statistics
##
##                Accuracy : 0.9906
##                  95% CI : (0.9882, 0.9926)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9881
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9842   0.9905   0.9774   0.9951
## Specificity            0.9964   0.9986   0.9951   0.9986   0.9994
## Pos Pred Value         0.9911   0.9940   0.9769   0.9929   0.9972
## Neg Pred Value         0.9998   0.9962   0.9980   0.9956   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1904   0.1727   0.1602   0.1829
## Detection Prevalence   0.2869   0.1916   0.1768   0.1614   0.1834
## Balanced Accuracy      0.9980   0.9914   0.9928   0.9880   0.9973
```

6: Prediction

Once selected the best model, is the time to apply it to the original test dataset.

```
actual_predict <- predict(modelrfcv, test_data3)
actual_predict
```

```
##  [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
```

Out-of-sample Error

OOS error could be measured with datasets that contain actual outcomes in order to do the comparison, which means that it is a post-mortem process; as a source to get lesson learned and adjust the model could works (Our test data -20 rows- does not contain the actual classe). In other words, we are making a decision based on in-sample error.