Practical Machine Learning Project

MBIYA-NGANDU LUBOYA

Sunday, September 21, 2014

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 (see the section on the Weight Lifting Exercise Dataset).

TODO

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 . Why you made the choices you did. . You will also use your prediction model to predict 20 different test cases.

Approach

We will try two mdifferent model and afterwards choose the one gives the best accuracy between . Recursive Partitioning and Regression Trees . Random Forest

We are expecting to get an out of sample error of about 0.5 % ##Data processing

```
library(caret)
```

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

```
training <- read.csv("pml-training.csv",na.strings=c("", "NA", "#DIV/0!"))

# For test
testing <- read.csv("pml-testing.csv",na.strings=c("", "NA", "#DIV/0!"))
```

Thowing some meaningless columns to avoid noise in the model (Columns that are not relevant for the model and those having more than 90% of NAs), such as:

. X, user_name, cvtd_timestamp , raw_timestamp_part_1 , raw_timestamp_part_2, new_window , . . .

We now get a dataset with 19622 rows and 54 columns.

For crossvalidation purpose, let divide the training dataset into subtraining set and subtraining_test

```
inTrain <- createDataPartition(y = training$classe, p = 0.8, list = FALSE)
subTraining <- training[inTrain,]
subTraining_test <- training[-inTrain,]
```

Model Construction

```
modFit1 <- train(subTraining$classe ~ ., data = subTraining, method = "rpart")
```

```
## Loading required package: rpart
```

```
modFit1
```

```
## CART
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
##
## Resampling results across tuning parameters:
##
##   cp    Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.04  0.5       0.40   0.03         0.05
##   0.06  0.4       0.17   0.05         0.09
##   0.11  0.3       0.06   0.04         0.06
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03858.
```

As we can see, the model as 61.2 % of accuracy. Let's consider another algorithm (Rf) and compare the result

```
tCtrl <- trainControl(method = "cv", number = 4)
modFit2 <- train(subTraining$classe ~ ., data = subTraining, method = "rf", prof = TRUE, trControl = tC
```

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

```
modFit2
```

```
## Random Forest
##
## 15699 samples
##    53 predictor
```

```
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 11774, 11774, 11774, 11775
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.001        0.001
##   27    1         1      0.001        0.001
##   53    1         1      0.003        0.003
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

As we can see, this model provides us an accuracy of 99.7 %, which is very intresting ! It's the one we will use, but let's first crossvalidate it on the subTraining_test and testing dataset to see the behavior.

Crossvalidation of the model

Prediction on Subtraining-test

```
subTrainingPredict <- predict(modFit2, subTraining_test)
confusionMatrix(subTrainingPredict, subTraining_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    2    0    0    0
##          B    0  757    1    0    0
##          C    0    0  683    3    0
##          D    0    0    0  640    0
##          E    1    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.996, 0.999)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.998
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.997    0.999    0.995    1.000
```

```
## Specificity             0.999    1.000    0.999    1.000    1.000
## Pos Pred Value          0.998    0.999    0.996    1.000    0.999
## Neg Pred Value          1.000    0.999    1.000    0.999    1.000
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.163    0.184
## Detection Prevalence    0.285    0.193    0.175    0.163    0.184
## Balanced Accuracy       0.999    0.999    0.999    0.998    1.000
```

As we can deduct on the confusion matrix, the out of sample error is 0.2039 % It is less than what we expected to have 0.5 %. This suggest that the model is optimistically good than the assumption made in approach section.

Prediction on the testing dataset

```
testingPredict <- predict(modFit2, testing)
testingPredict
```

```
##  [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
```