Practical Machine Learning project report

Tom Zumer

July 01 2018

Overview

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.

Load data

We load the data and do some basic analisys to get a grasp on the data.

```
library(caret)
library(rattle)
library(randomForest)
library(rpart)
library(parallel)
library(doParallel)

pml_training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),
                  na.strings = c("NA","","#DIV/0!"), header = TRUE)
pml_testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),
                 na.strings = c("NA","","#DIV/0!"), header = TRUE)
```

```
dim(pml_testing)
```

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

```
dim(pml_training)
```

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

```
head(pml_training)
str(pml_training)
```

```
unique(pml_training$classe)
```

```
## [1] A B C D E
## Levels: A B C D E
```

```
t <- table(pml_training$classe)
t
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Cleaning data

We need to clean the data because we need to remove all NA comlumn values for our model. We also remove columns with low variance.

```r
nas <- is.na(pml_training)
pml_training <- pml_training[,colSums(nas) == 0]
pml_testing <- pml_testing[,colSums(nas) == 0]

zeros <- nearZeroVar(pml_training)
pml_training <- pml_training[,-zeros]
pml_testing <- pml_testing[,-zeros]
pml_training <- pml_training[,-(1:5)]
pml_testing <- pml_testing[,-(1:5)]
pml_testing$problem_id <- NULL
```

Train and predict data

We divide data into 60% for training set and 40% for testing set.

```r
set.seed(42)
inTrain <- createDataPartition(pml_training$classe, p = 0.6, list = FALSE)
training <- pml_training[inTrain,]
testing <- pml_training[-inTrain,]
```

We will predict variable classes with the following methods: Decision Tree, Generalized Boosted Regression and Random Forest. Based on prediction accuracies we will select a model with the highest one and pick cross validation to define resampling schema.

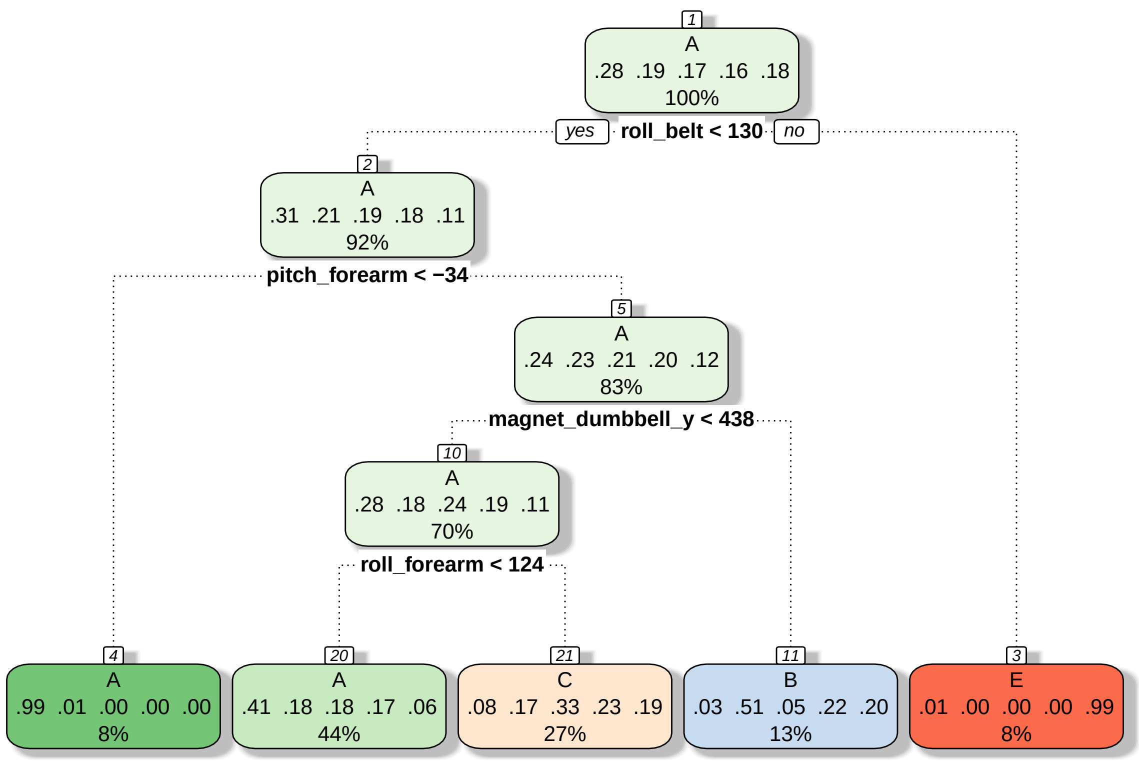
Decision Tree model

```r
trainCont <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
modfitDt <- train(classe ~ .,data = training, method = "rpart", trControl = trainCont)
predDt = predict(modfitDt,testing)
confusionMatrix(predDt, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2024  625  644  573  230
##          B   41  515   39  245  188
##          C  161  378  685  468  370
##          D    0    0    0    0    0
##          E    6    0    0    0  654
##
## Overall Statistics
##
##                Accuracy : 0.4943
##                  95% CI : (0.4831, 0.5054)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3388
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068  0.33926  0.50073   0.0000  0.45354
## Specificity            0.6309  0.91893  0.78743   1.0000  0.99906
## Pos Pred Value         0.4941  0.50097  0.33220      NaN  0.99091
## Neg Pred Value         0.9445  0.85289  0.88192   0.8361  0.89034
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2580  0.06564  0.08731   0.0000  0.08335
## Detection Prevalence   0.5220  0.13102  0.26281   0.0000  0.08412
## Balanced Accuracy      0.7689  0.62910  0.64408   0.5000  0.72630
```

```r
fancyRpartPlot(modfitDt$finalModel)
```



Rattle 2018−jul.−02 20:55:43 Goku

Generalized Boosted Regression Model

```r
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
modfitGbm <- train(classe ~ .,data = training, method = "gbm", trControl = trainCont)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2396
##      2        1.4577             nan     0.1000    0.1597
##      3        1.3570             nan     0.1000    0.1397
##      4        1.2718             nan     0.1000    0.1071
##      5        1.2047             nan     0.1000    0.0907
```

```
##      6        1.1477             nan     0.1000    0.0789
##      7        1.0984             nan     0.1000    0.0797
##      8        1.0488             nan     0.1000    0.0674
##      9        1.0064             nan     0.1000    0.0639
##     10        0.9663             nan     0.1000    0.0699
##     20        0.6998             nan     0.1000    0.0279
##     40        0.4594             nan     0.1000    0.0121
##     60        0.3342             nan     0.1000    0.0119
##     80        0.2534             nan     0.1000    0.0052
##    100        0.1986             nan     0.1000    0.0039
##    120        0.1555             nan     0.1000    0.0021
##    140        0.1266             nan     0.1000    0.0016
##    150        0.1138             nan     0.1000    0.0022
```

```r
stopCluster(cluster)
registerDoSEQ()
predGbm = predict(modfitGbm,testing)
confusionMatrix(predGbm, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   18    0    0    0
##          B    4 1489    7   11    6
##          C    0   10 1356   22    2
##          D    1    1    5 1253   15
##          E    0    0    0    0 1419
##
## Overall Statistics
##
##                Accuracy : 0.987
##                  95% CI : (0.9842, 0.9894)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9836
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9809   0.9912   0.9743   0.9840
## Specificity            0.9968   0.9956   0.9948   0.9966   1.0000
## Pos Pred Value         0.9920   0.9815   0.9755   0.9827   1.0000
## Neg Pred Value         0.9991   0.9954   0.9981   0.9950   0.9964
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1898   0.1728   0.1597   0.1809
## Detection Prevalence   0.2861   0.1933   0.1772   0.1625   0.1809
## Balanced Accuracy      0.9973   0.9882   0.9930   0.9855   0.9920
```

Random Forest model

```r
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
modfitRf <- train(classe ~ .,data = training, method = "rf", trControl = trainCont)
stopCluster(cluster)
registerDoSEQ()
predRf = predict(modfitRf,testing)
confusionMatrix(predRf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    5    0    0    0
##          B    1 1513    1    0    2
##          C    0    0 1367   10    0
##          D    0    0    0 1275    7
##          E    0    0    0    1 1433
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9967   0.9993   0.9914   0.9938
## Specificity            0.9991   0.9994   0.9985   0.9989   0.9998
## Pos Pred Value         0.9978   0.9974   0.9927   0.9945   0.9993
## Neg Pred Value         0.9998   0.9992   0.9998   0.9983   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1928   0.1742   0.1625   0.1826
## Detection Prevalence   0.2850   0.1933   0.1755   0.1634   0.1828
## Balanced Accuracy      0.9993   0.9980   0.9989   0.9952   0.9968
```

Based on prediction accuracy on all three models we can conclude that the best model for our purpuse is Random Forest with accuracy of 0.9966 and Out of Sample Error of 0.0034. But with such high accuracy we can suspect that the model is overfitting.

Prediction with test data

We now make prediction with our model on the original test data.

```r
finalpred <- predict(modfitRf, pml_testing)
finalpred
```

```
##  [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
```