Practical Machine Learning

Lalith Kumar Vemali

6/20/2019

#Executive Summary

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

In this report we show how to deal with this information from reading data, cleaning, explore and building a machine learning algorithm to predict if the person how weel is a person sitting-down, standing-up, standing, walking or sitting (5 different classes how define de quality of the exercise). In the modeling part of this study a random forest was built and 10 fold cross-validation was used to fine tune the parameters of the model. A high level of accuracy was achieved in the test data so we expect this model to generalize very well.

Preloading packages

Downloading and reading the files.

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pmltraining <- read.csv(url(trainUrl),sep = ",", na.strings = c("", "NA"))
pmltesting <- read.csv(url(testUrl),sep = ",", na.strings = c("", "NA"))
```

Testing the files for my start point.

```
dim(pmltraining)
```

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

```
dim(pmltesting)
```

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

Pre screening of variables with too many NA values

```
training.nonNAs <- pmltraining[ , colSums(is.na(pmltraining)) == 0]
dim(training.nonNAs)
```

```
## [1] 19622    60
```

Cleaning my values

```
cleanpmlTraining<-training.nonNAs[,-c(1:8)]
dim(cleanpmlTraining)
```

```
## [1] 19622    52
```

```
cleanpmltesting<-pmltesting[,names(cleanpmlTraining[,-52])]
dim(cleanpmltesting)
```

```
## [1] 20 51
```

Partitioning the data to create a 75% training set and a 25% test set.

```
inTrain<-createDataPartition(y=cleanpmlTraining$classe, p=0.75,list=F)
training<-cleanpmlTraining[inTrain,]
test<-cleanpmlTraining[-inTrain,]
dim(training)
```

```
## [1] 14718    52
```

Cross validation using a random forest done at 5 fold.This achieves 95% CI(0.9906,0.9954),Accuracy 99% and a kappa value of 0.992

```
Modfit1<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rffit<-train(classe~.,data=training, method="rf", trControl=Modfit1, verbose=F)
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
pred.rf<-predict(rffit, newdata=test)
confusionMatrix(pred.rf, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
```

```
##          B    0  941    6    0    0
##          C    0    2  843   10    0
##          D    0    0    6  793    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9916   0.9860   0.9863   0.9978
## Specificity            0.9983   0.9985   0.9970   0.9980   0.9998
## Pos Pred Value         0.9957   0.9937   0.9860   0.9900   0.9989
## Neg Pred Value         1.0000   0.9980   0.9970   0.9973   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1919   0.1719   0.1617   0.1833
## Detection Prevalence   0.2857   0.1931   0.1743   0.1633   0.1835
## Balanced Accuracy      0.9991   0.9950   0.9915   0.9922   0.9988
```

```
pred.20<-predict(rffit, newdata=cleanpmltesting)
pred.20
```

```
##  [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
```

using Fancy rpart plot

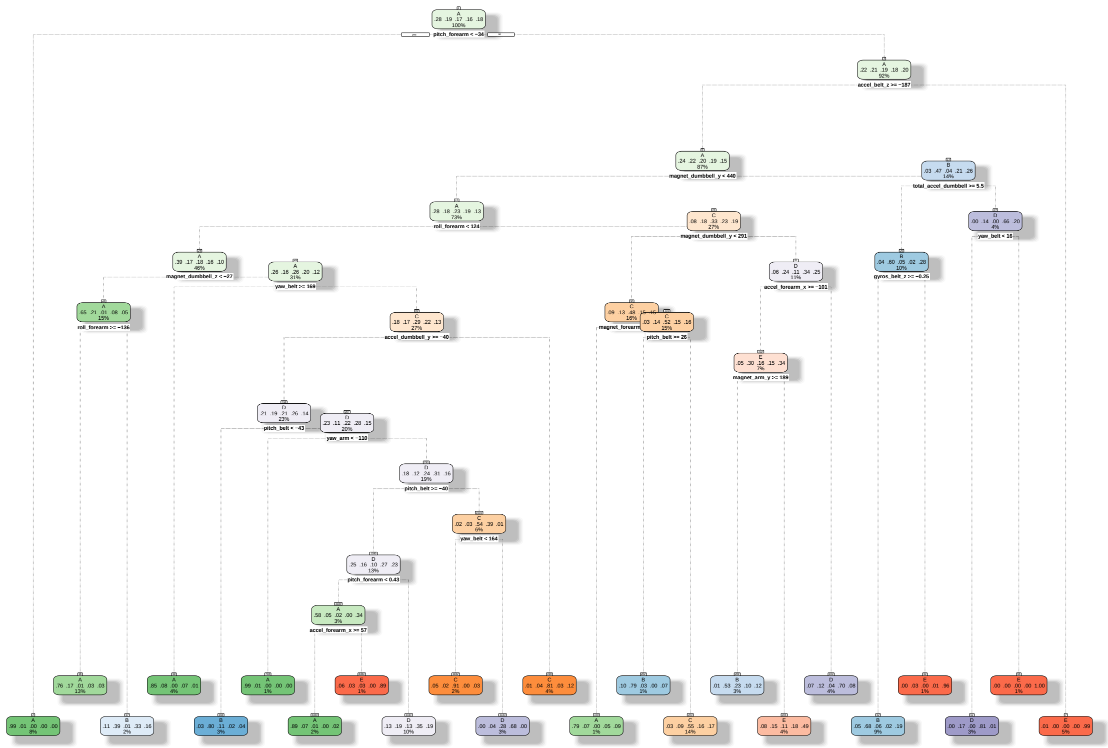
```
set.seed(1234)
modFit2 <- rpart(classe ~ ., data=cleanpmlTraining, method="class")
print(modFit2)
```

```
## n= 19622
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 19622 14042 A (0.28 0.19 0.17 0.16 0.18)
##      2) pitch_forearm< -33.95 1578    10 A (0.99 0.0063 0 0 0) *
##      3) pitch_forearm>=-33.95 18044 14032 A (0.22 0.21 0.19 0.18 0.2)
##        6) accel_belt_z>=-187.5 17009 13003 A (0.24 0.22 0.2 0.19 0.15)
##         12) magnet_dumbbell_y< 439.5 14253 10328 A (0.28 0.18 0.23 0.19 0.13)
##           24) roll_forearm< 123.5 8980  5460 A (0.39 0.17 0.18 0.16 0.095)
##             48) magnet_dumbbell_z< -27.5 2968  1029 A (0.65 0.21 0.013 0.076 0.05)
##               96) roll_forearm>=-136.5 2478   591 A (0.76 0.17 0.013 0.026 0.028) *
##               97) roll_forearm< -136.5 490   297 B (0.11 0.39 0.01 0.33 0.16) *
##             49) magnet_dumbbell_z>=-27.5 6012  4431 A (0.26 0.16 0.26 0.2 0.12)
##               98) yaw_belt>=168.5 750   114 A (0.85 0.079 0.0013 0.067 0.0053) *
```

```
##                 99) yaw_belt< 168.5 5262  3714 C (0.18 0.17 0.29 0.22 0.13)
##                  198) accel_dumbbell_y>=-40.5 4521  3368 D (0.21 0.19 0.21 0.26 0.14)
##                    396) pitch_belt< -42.85 520   104 B (0.031 0.8 0.11 0.025 0.038) *
##                    397) pitch_belt>=-42.85 4001  2861 D (0.23 0.11 0.22 0.28 0.15)
##                      794) yaw_arm< -110.5 267     4 A (0.99 0.015 0 0 0) *
##                      795) yaw_arm>=-110.5 3734  2594 D (0.18 0.12 0.24 0.31 0.16)
##                       1590) pitch_belt>=-40.45 2581  1893 D (0.25 0.16 0.1 0.27 0.23)
##                         3180) pitch_forearm< 0.425 637   265 A (0.58 0.055 0.017 0 0.34)
##                           6360) accel_forearm_x>=56.5 400    43 A (0.89 0.072 0.012 0 0.022) *
##                           6361) accel_forearm_x< 56.5 237    27 E (0.063 0.025 0.025 0 0.89) *
##                         3181) pitch_forearm>=0.425 1944  1256 D (0.13 0.19 0.13 0.35 0.19) *
##                       1591) pitch_belt< -40.45 1153   526 C (0.023 0.03 0.54 0.39 0.011)
##                         3182) yaw_belt< 163.5 488    46 C (0.047 0.018 0.91 0.002 0.027) *
##                         3183) yaw_belt>=163.5 665   214 D (0.0045 0.039 0.28 0.68 0) *
##                  199) accel_dumbbell_y< -40.5 741   143 C (0.0081 0.04 0.81 0.028 0.12) *
##             25) roll_forearm>=123.5 5273  3546 C (0.077 0.18 0.33 0.23 0.19)
##               50) magnet_dumbbell_y< 290.5 3093  1615 C (0.091 0.13 0.48 0.15 0.15)
##                100) magnet_forearm_z< -251 238    49 A (0.79 0.071 0 0.046 0.088) *
##                101) magnet_forearm_z>=-251 2855  1377 C (0.033 0.14 0.52 0.15 0.16)
##                  202) pitch_belt>=26.15 189    39 B (0.1 0.79 0.032 0 0.074) *
##                  203) pitch_belt< 26.15 2666  1194 C (0.028 0.09 0.55 0.16 0.17) *
##               51) magnet_dumbbell_y>=290.5 2180  1430 D (0.056 0.24 0.11 0.34 0.25)
##                102) accel_forearm_x>=-101.5 1398   923 E (0.051 0.3 0.16 0.15 0.34)
##                  204) magnet_arm_y>=188.5 573   267 B (0.014 0.53 0.23 0.1 0.12) *
##                  205) magnet_arm_y< 188.5 825   420 E (0.076 0.15 0.11 0.18 0.49) *
##                103) accel_forearm_x< -101.5 782   237 D (0.066 0.12 0.036 0.7 0.077) *
##           13) magnet_dumbbell_y>=439.5 2756  1470 B (0.029 0.47 0.039 0.21 0.26)
##             26) total_accel_dumbbell>=5.5 1948   774 B (0.042 0.6 0.054 0.019 0.28)
##               52) gyros_belt_z>=-0.255 1721   554 B (0.047 0.68 0.062 0.02 0.19) *
##               53) gyros_belt_z< -0.255 227    10 E (0 0.031 0 0.013 0.96) *
##             27) total_accel_dumbbell< 5.5 808   277 D (0 0.14 0.0025 0.66 0.2)
##               54) yaw_belt< 16.475 652   121 D (0 0.17 0.0031 0.81 0.011) *
##               55) yaw_belt>=16.475 156     0 E (0 0 0 0 1) *
##          7) accel_belt_z< -187.5 1035     7 E (0.0058 0.00097 0 0 0.99) *
```

```r
fancyRpartPlot(modFit2, digits=2)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2019−Jun−20 14:13:59 lalithkumar.vemali

Predicting test data set

result <-predict(rffit,cleanpmltesting[, -length(names(cleanpmltesting))])

The accuracy achieved with rpart plot was less due to overplotting.The random forest method is the best fit model.