Practical Machine Learning

ydong

May 20, 2019

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which people did the exercise. Include the model building, cross validation and use the model to predict 20 more different test cases.

Package

```
library(caret)
library(rattle)
library(caret)
library(rpart)
library(randomForest)
library(gbm)
```

Load data

```
train <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header = T)
test<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header = T)
dim(train);dim(test)
```

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

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

The training dataset includes 160 variables, 19622 observation. Here we ignore the varialbes with missing values for now.

Data cleaning

```
#remove variables that contains missing values
#remove the first seven varialbes
training <- train[,colSums(is.na(train))==0]
training <- training[,-c(1:7)]
dim(training)
```

```
## [1] 19622    86
```

```
# Repeat for the test set
testing <- test[,colSums(is.na(test))==0]
testing <- testing[,-c(1:7)]
dim(testing)
```

```
## [1] 20 53
```

Preparing the datasets for prediction Preparing the data for prediction by splitting the training data into 70% as train data and 30% as test data. This splitting helps to compute the out-of-sample errors.

```
#Data slicing to training and testing dataset
set.seed(7777777)
intrain <- createDataPartition(training$classe, p=0.7, list=F)
train1 <- training[intrain,]
test1 <- training[-intrain,]
dim(train1);dim(test1)
```

```
## [1] 13737    86
```

```
## [1] 5885   86
```

```
#Cleaning further by removing the variables that are near-zero-variance

nzv <- nearZeroVar(training)
train1 <- train1[, -nzv]
test1  <- test1[, -nzv]
dim(train1);dim(test1)
```

```
## [1] 13737    53
```

```
## [1] 5885   53
```

Here we use the findCorrelation function to search for highly correlated attributes with a cut off of 0.8

```
cor_matrix <- cor(train1[, -53])
#remove the y variable for correlation matirx
hcr = findCorrelation(cor_matrix, cutoff=0.8)
names(train1)[hcr]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
##  [4] "accel_dumbbell_z" "accel_belt_x"     "pitch_belt"
##  [7] "accel_dumbbell_x" "accel_arm_x"      "magnet_arm_y"
## [10] "gyros_forearm_y"  "gyros_dumbbell_x" "gyros_dumbbell_z"
## [13] "gyros_arm_x"
```

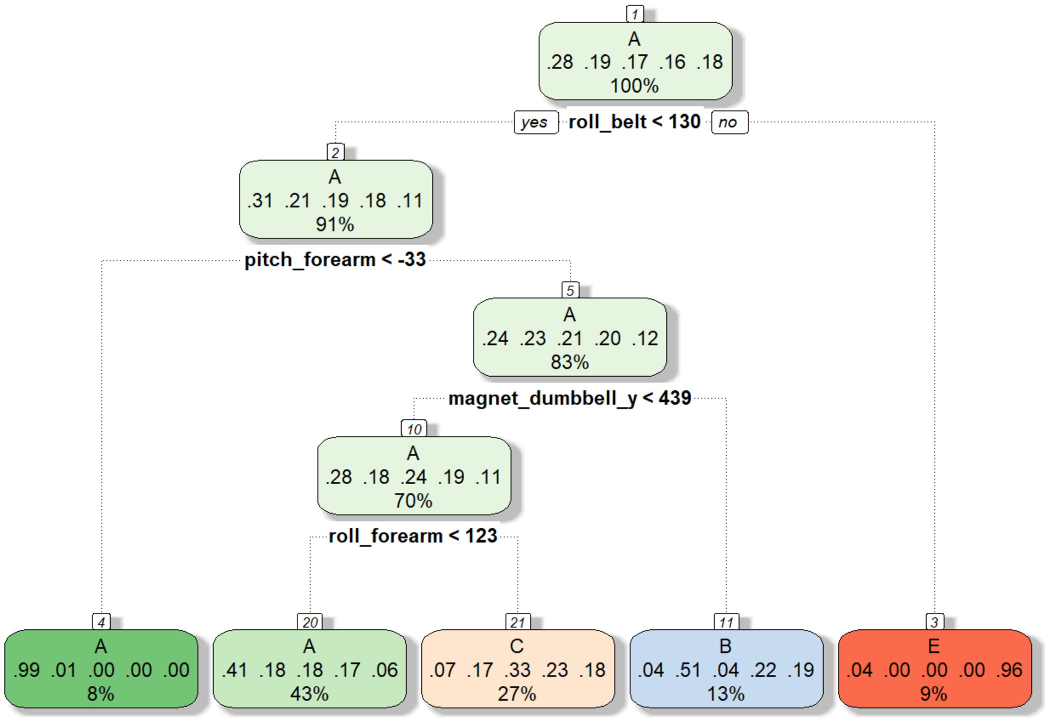
Model building

For this project,classification trees and random forests and boosing are applied to predict the outcome.

Train with classification tree

```
library(rpart)
trcontrol<- trainControl(method="cv", number=3,verboseIter = F)
model.ct <- train(classe~., data=train1, method="rpart", trControl=trcontrol)

fancyRpartPlot(model.ct$finalModel)
```



Rattle 2019-May-23 21:35:07 dongy

```
pred.ct <- predict(model.ct,newdata=test1)
confm.ct<- confusionMatrix(test1$classe,pred.ct)

# display confusion matrix and model accuracy
confm.ct$table;confm.ct$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1498   23  128    0   25
##          B  473  372  294    0    0
##          C  495   33  498    0    0
##          D  444  177  343    0    0
##          E  158  136  269    0  519
```

```
##  Accuracy
## 0.4905692
```

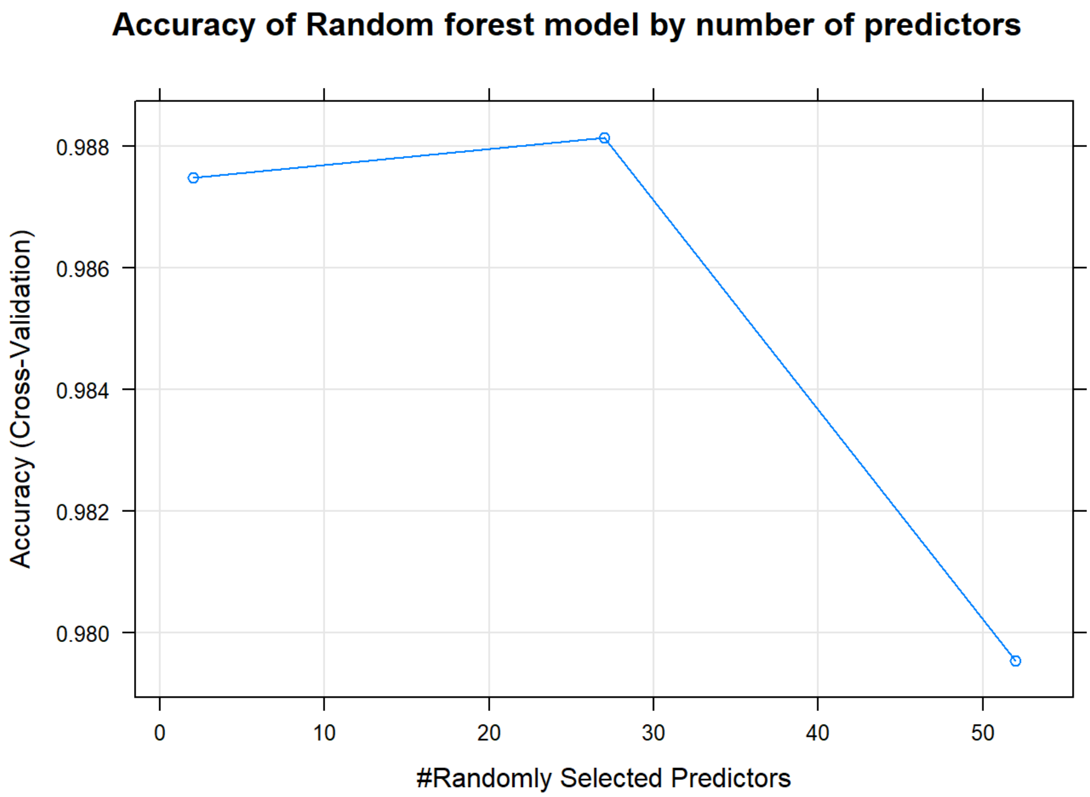
The Accuracy is below 0.5, suggesting the model is not good enough.

Train with random forests

```
model.rf <- train(classe~., data=train1, method="rf", trControl=trcontrol)
print(model.rf)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9157, 9158, 9159
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9874791  0.9841586
##   27    0.9881341  0.9849893
##   52    0.9795438  0.9741216
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
plot(model.rf,main="Accuracy of Random forest model by number of predictors")
```




```
pred.rf <- predict(model.rf,newdata=test1)

confmrf <- confusionMatrix(test1$classe,pred.rf)

# display confusion matrix and model accuracy
confmrf$table;confmrf$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    1    0    0
##          B    9 1126    4    0    0
##          C    0    1 1023    2    0
##          D    0    0   12  952    0
##          E    0    2    0    6 1074
```

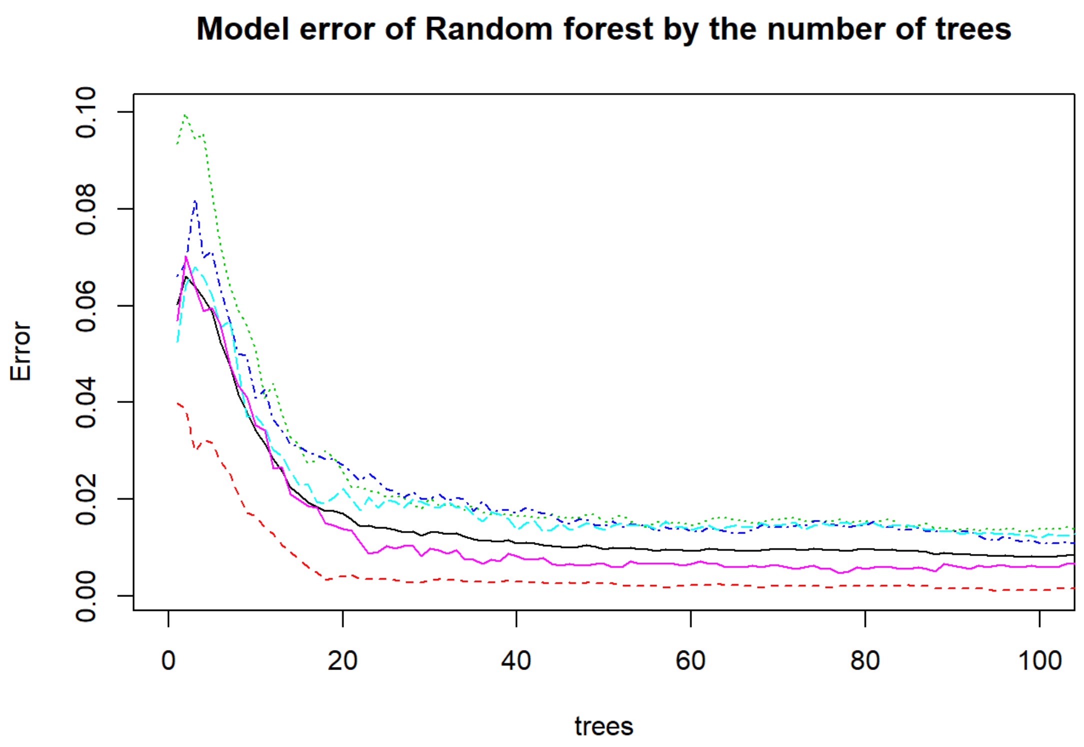
```
##  Accuracy
## 0.9935429
```

The accuracy rate using the random forest is 0.993,it might be overfitting.

```
model.rf$finalModel$classes
```

```
## [1] "A" "B" "C" "D" "E"
```

```
plot(model.rf$finalModel,main="Model error of Random forest by the number of trees",xlim=c(0,100))
```




```
# Compute the variable importance
mostimpvar<- varImp(model.rf)
mostimpvar
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          61.39
## yaw_belt               54.53
## roll_forearm           45.20
## magnet_dumbbell_y      45.09
## pitch_belt             44.86
## magnet_dumbbell_z      43.83
## accel_dumbbell_y       23.38
## accel_forearm_x        17.78
## roll_dumbbell          16.56
## magnet_dumbbell_x      16.46
## magnet_belt_z          15.06
## accel_belt_z           14.82
## magnet_forearm_z       14.42
## accel_dumbbell_z       14.06
## total_accel_dumbbell   12.44
## gyros_belt_z           11.32
## yaw_arm                11.27
## magnet_belt_y          10.73
## magnet_belt_x           9.91
```

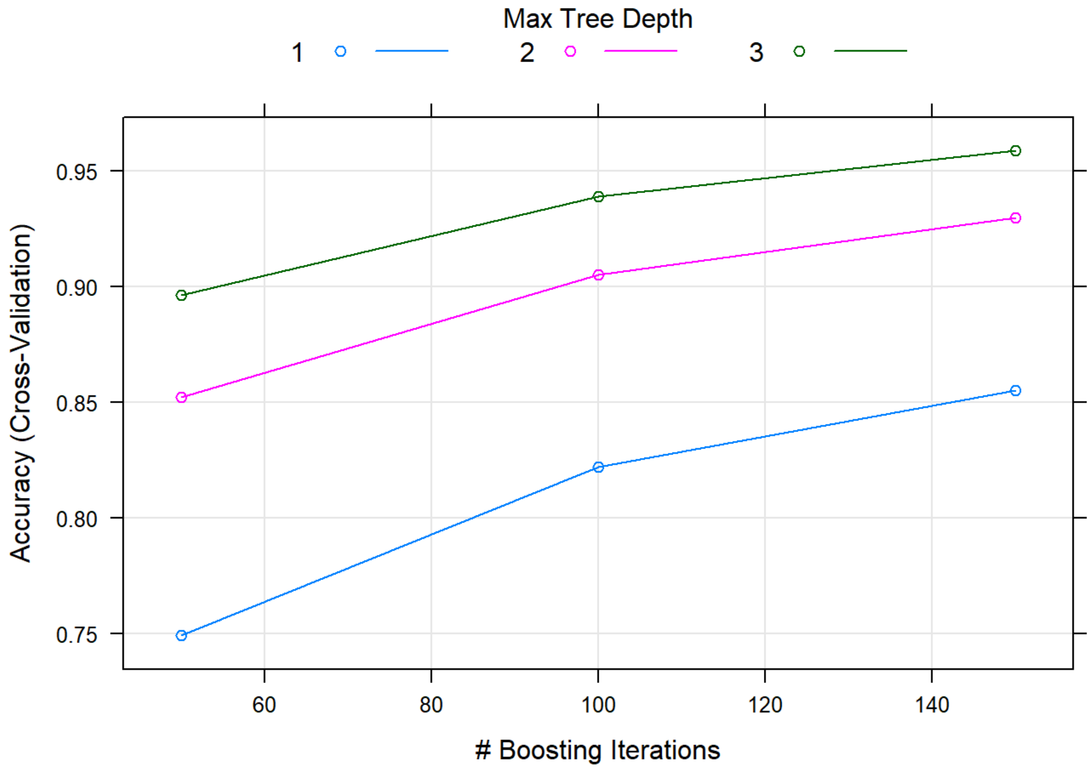
```
#only show the 20 most important variables
```

Train with boosting method

```
model.bt<- train(classe~., data=train1, method="gbm", trControl=trcontrol, verbose=F)
print(model.bt)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7492902  0.6820257
##   1                  100      0.8220135  0.7747453
##   1                  150      0.8552814  0.8168569
##   2                   50      0.8522239  0.8127796
##   2                  100      0.9052195  0.8800428
##   2                  150      0.9298246  0.9112008
##   3                   50      0.8962656  0.8686607
##   3                  100      0.9387785  0.9225274
##   3                  150      0.9585062  0.9475063
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot(model.bt)
```




```
pred.bt <- predict(model.bt,newdata=test1)

confm.bt <- confusionMatrix(test1$classe,pred.bt)

confm.bt$table;confm.bt$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1648   15    7    4    0
##          B   47 1059   31    2    0
##          C    0   28  986   10    2
##          D    1    2   40  914    7
##          E    3    8    5    9 1057
```

```
##  Accuracy
## 0.9624469
```

The accuracy is 0.959, therefore the out-of-sample error is 0.041

```
finalmodel <- predict(model.bt,newdata=testing)
finalmodel
```

```
##  [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
```