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.

```
library(caret)
library(rattle)
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)]

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

dim(training);dim(testing)
```

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

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

```
#Cleaning further by removing the variables that are near-zero-variance
nzv <- nearZeroVar(training)
train1 <- training[, -nzv]
test1  <- testing[, -nzv]
dim(train1);dim(test1)
```

```
## [1] 19622    53
```

```
## [1] 20 29
```

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(7654321)
intrain <- createDataPartition(training$classe, p=0.7, list=F)
train1 <- train1[intrain,]
test1 <- train1[-intrain,]
dim(train1);dim(test1)
```

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

```
## [1] 4119   53
```

Here we use the findCorrelation function to show the variables with >0.8 correlation

```
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"
```

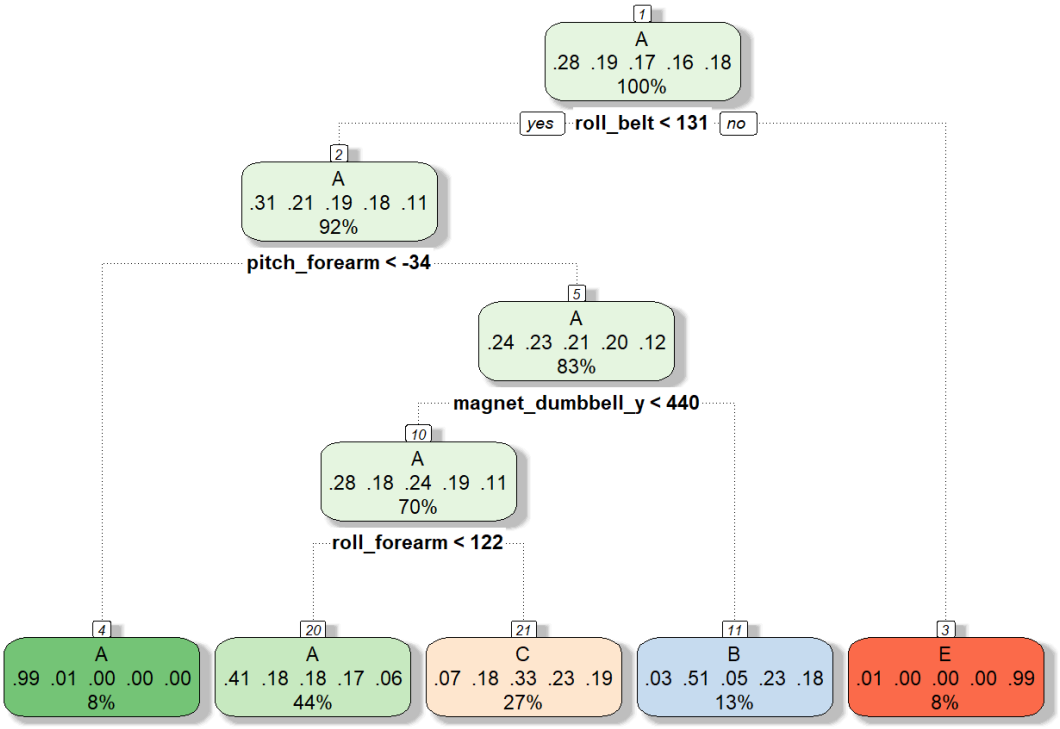
Model building

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 22:28:15 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 1078   12   84    0    2
##         B  323  287  194    0    0
##         C  307   21  379    0    0
##         D  285  127  273    0    0
##         E   97   77  205    0  368
```

```
##  Accuracy
## 0.5127458
```

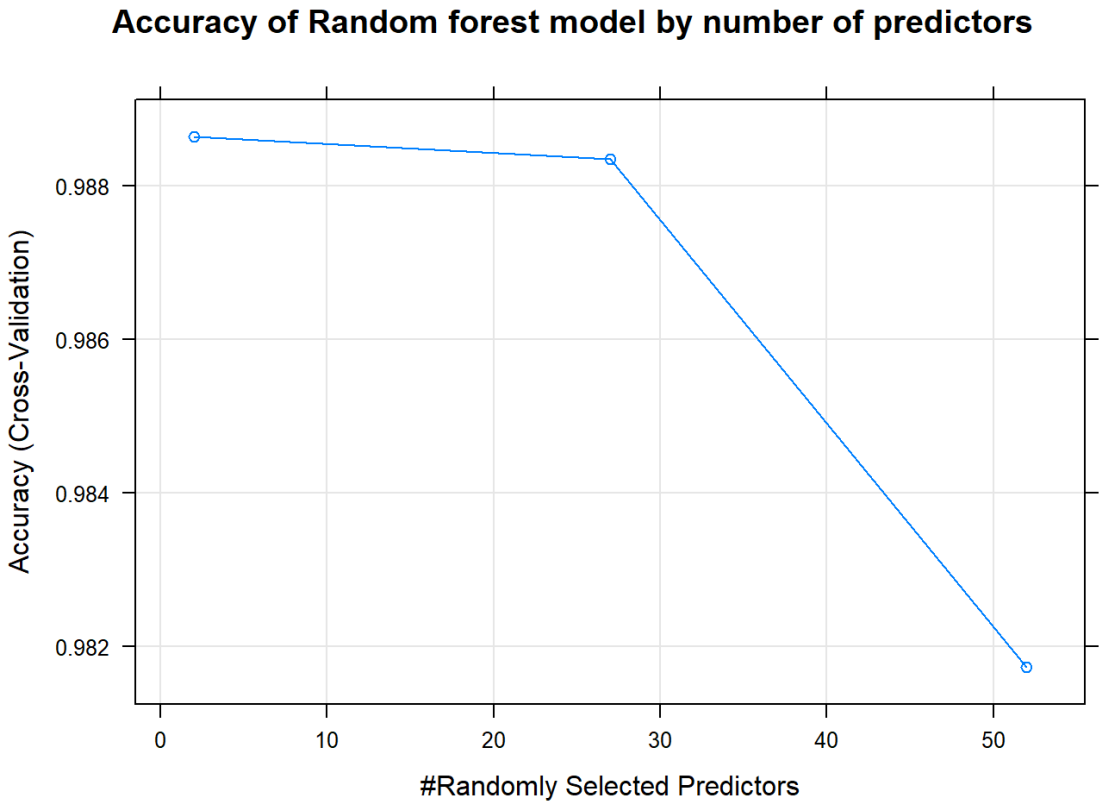
The accuracy is below 0.513, the out-of sample error is about 0.487, suggesting the model is not good enough.

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, 9159, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9886442  0.9856324
##   27    0.9883529  0.9852646
##   52    0.9817286  0.9768852
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
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 1176    0    0    0    0
##          B    0  804    0    0    0
##          C    0    0  707    0    0
##          D    0    0    0  685    0
##          E    0    0    0    0  747
```

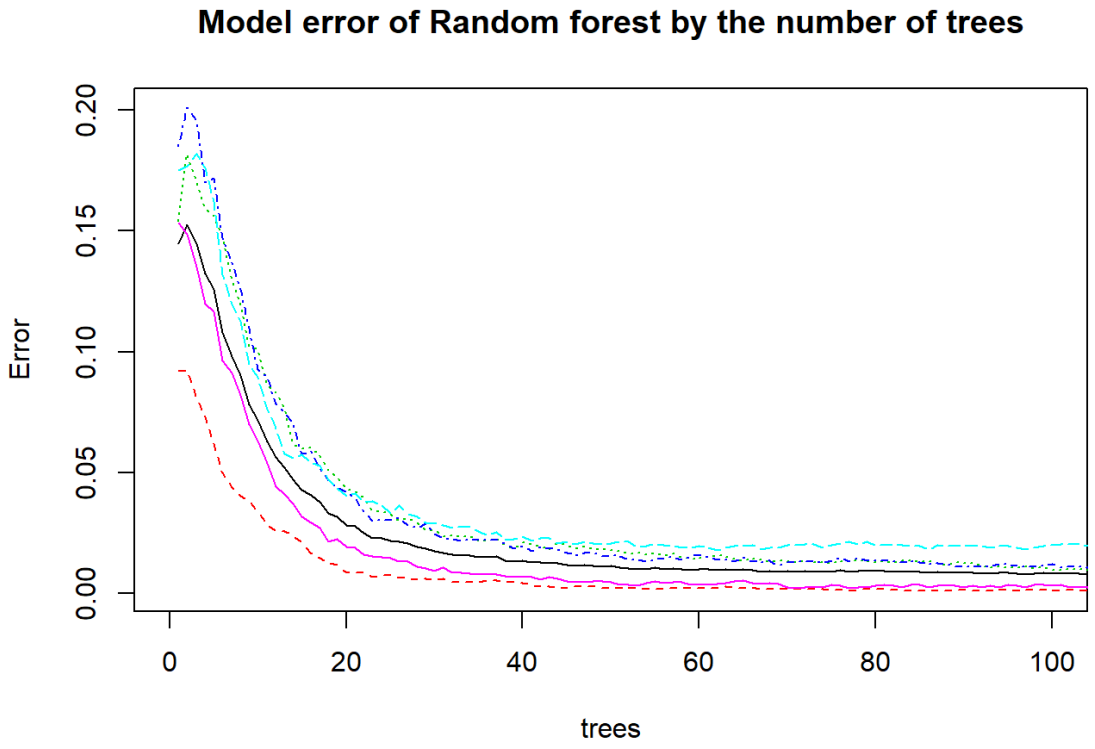
```
## Accuracy
##        1
```

The accuracy rate using the random forest is 1,the out-of-sample error is about 0. We could understand that 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
## yaw_belt              85.41
## magnet_dumbbell_z     74.20
## pitch_belt            67.14
## pitch_forearm         66.93
## magnet_dumbbell_y     66.59
## roll_forearm          56.49
## magnet_dumbbell_x     55.46
## accel_belt_z          48.04
## magnet_belt_z         46.74
## accel_dumbbell_y      45.83
## roll_dumbbell         45.83
## magnet_belt_y         44.18
## accel_dumbbell_z      39.14
## roll_arm              38.31
## accel_forearm_x       33.59
## yaw_dumbbell          31.74
## accel_dumbbell_x      31.30
## magnet_arm_x          30.06
## total_accel_dumbbell  30.03
```

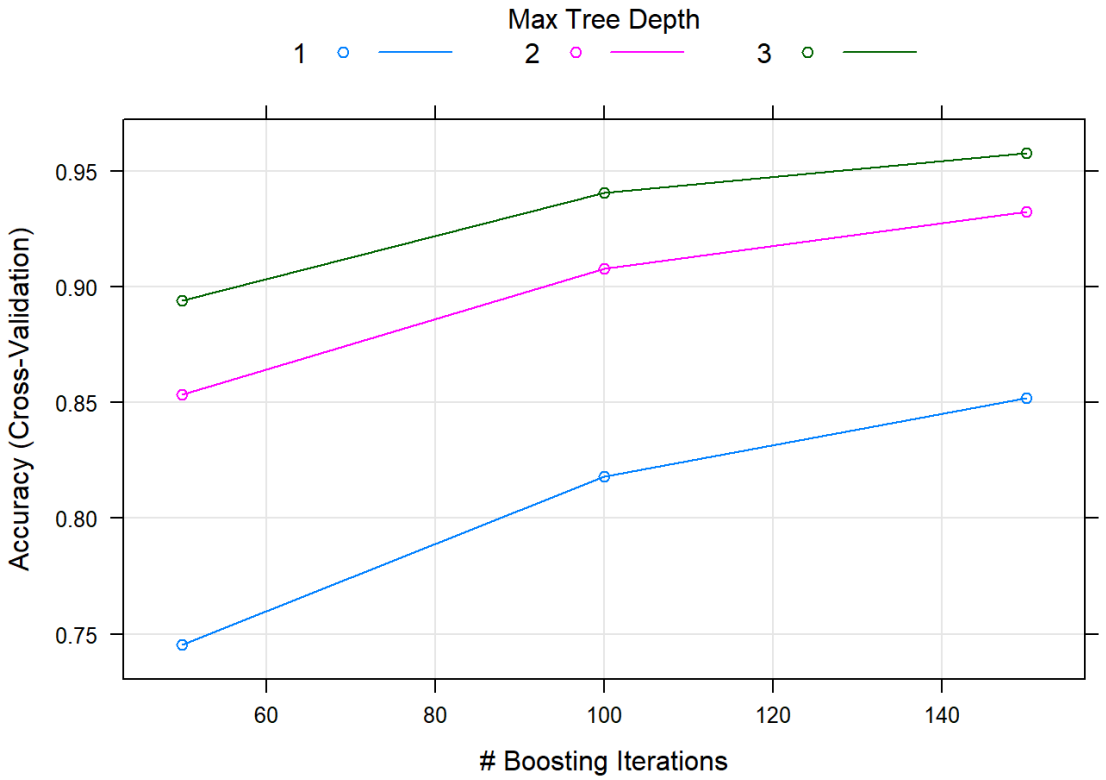
```
#only show the 20 most important variables
```

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: 9159, 9157, 9158
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7452852  0.6770756
##   1                  100      0.8180100  0.7696530
##   1                  150      0.8519333  0.8126482
##   2                   50      0.8533161  0.8141740
##   2                  100      0.9077679  0.8832819
##   2                  150      0.9325909  0.9147169
##   3                   50      0.8940820  0.8659455
##   3                  100      0.9407443  0.9250263
##   3                  150      0.9577057  0.9464934
##
## 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 1161    6    5    4    0
##          B   15  776   12    1    0
##          C    0   10  690    5    2
##          D    0    0   13  671    1
##          E    0    8    2    4  733
```

```
##  Accuracy
## 0.9786356
```

The accuracy is 0.979, therefore the out-of-sample error is 0.021. This boosting model may perfom better on the testing dataset.

```
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
```