Practical ML Project

Adam Yao Guang Chen

28 January 2017

Table of Contents

## Introduction

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.

## Initialization

library(RCurl)

## Loading required package: bitops

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(ggplot2)  
library(rpart)  
library(rpart.plot)

## Data Processing

# Download data  
URL\_Train <-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
URL\_Test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
  
File\_Train <- "./data/pml-training.csv"  
File\_Test <- "./data/pml-testing.csv"  
  
if (!file.exists("./data")) {  
 dir.create("./data")  
}  
  
download.file(URL\_Train, File\_Train)  
download.file(URL\_Test, File\_Test)  
  
# Read Data  
TrainData = read.csv(File\_Train)  
TestData = read.csv(File\_Test)  
  
dim(TrainData)

## [1] 19622 160

dim(TestData)

## [1] 20 160

#names(TrainData)

There are 19622 & 20 observations in training & testing data respectively. The number of variables is 160. The "classe" variable in the training set is the outcome to predict.

## Data cleansing

1. Remove all columns with NA values in test dataset
2. Keep numeric columns & Keep Classes for TrainData
3. Remove columns not much ralevant to the final preditions

# remove columns with NA in TestData  
SelectedColumns = colSums(is.na(TestData)) == 0  
TrainData = TrainData[, SelectedColumns]  
TestData = TestData[, SelectedColumns]  
  
# Keep numeric columns & Keep Classes for TrainData  
classe <- TrainData$classe  
TrainData <- TrainData[, sapply(TrainData, is.numeric)]  
TestData <- TestData[, sapply(TestData, is.numeric)]  
TrainData$classe <- classe  
  
# Remove columns not much ralevant to the final preditions  
TrainData <- TrainData[, !grepl("^X|timestamp|window", names(TrainData))]  
TestData <- TestData[, !grepl("^X|timestamp|window", names(TestData))]  
  
dim(TrainData)

## [1] 19622 53

dim(TestData)

## [1] 20 53

There were 53 variables remained.

## Boosting Model

Fit model with boosting algorithm and 10-fold cross validation to predict "classe" by other variables.

set.seed(1104)  
Boosting <- train(classe ~ ., method = "gbm", data = TrainData, verbose = F, trControl = trainControl(method = "cv", number = 10))

## Loading required package: gbm

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: splines

## Loading required package: parallel

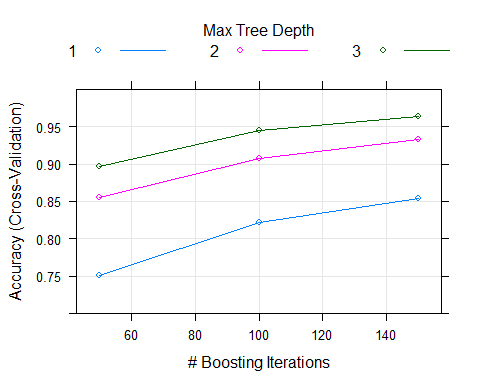
## Loaded gbm 2.1.1

## Loading required package: plyr

Boosting

## Stochastic Gradient Boosting   
##   
## 19622 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 17658, 17659, 17661, 17660, 17659, 17661, ...   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.7508913 0.6841149  
## 1 100 0.8222926 0.7751024  
## 1 150 0.8540928 0.8153621  
## 2 50 0.8553674 0.8167886  
## 2 100 0.9072475 0.8826251  
## 2 150 0.9333927 0.9157149  
## 3 50 0.8972079 0.8698564  
## 3 100 0.9445540 0.9298504  
## 3 150 0.9638683 0.9542902  
##   
## 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(Boosting, ylim = c(0.7, 1))



The boosting algorithm generated a very good model with accuracy of 96.39% and the estimated out-of-sample error is 4.61%.

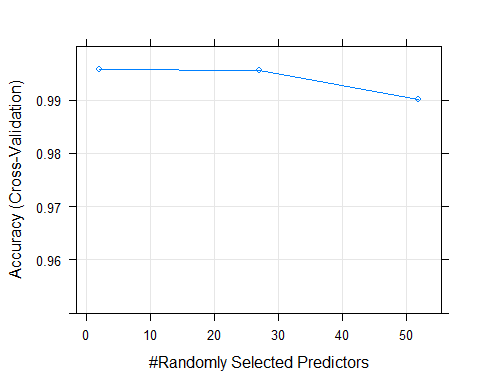
## Random Forests model

Fit model with random forests algorithm and 10-fold cross validation to predict "classe" by other variables.

set.seed(1104)  
RandomForests <- train(classe ~ ., method = "rf", data = TrainData, importance = T, trControl = trainControl(method = "cv", number = 10))  
  
#Plot accuracy of this model on the scale of [0.95, 1].   
RandomForests

## Random Forest   
##   
## 19622 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 17658, 17659, 17661, 17660, 17659, 17661, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9956683 0.9945207  
## 27 0.9954645 0.9942628  
## 52 0.9901642 0.9875574  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

plot(RandomForests, ylim = c(0.95, 1))



#The Random Forests algorithm generated an excellent model with accuracy of 99.567% and the estimated out-of-sample error is 0.433%.   
  
RandomForests$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry, importance = ..1)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 0.44%  
## Confusion matrix:  
## A B C D E class.error  
## A 5577 3 0 0 0 0.0005376344  
## B 11 3781 5 0 0 0.0042138530  
## C 0 17 3404 1 0 0.0052600818  
## D 0 0 42 3172 2 0.0136815920  
## E 0 0 0 5 3602 0.0013861935

# Random Foresters predictions  
(prediction <- as.character(predict(RandomForests, TestData)))

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

## Predictions with Random Forests

The random forests algorithm generated a more accurate model with accuracy close to 1. So we'll choose Random Forests model for prediction.

The final line is predit results of test data.