Practical Machine Learning Course Project

Siddhi Lal Kayestha

May 31, 2016

Executive Summary

Data collection about personal activity has been more easy with the availability of devices such as Jawbone Up, Nike FuelBand, and Fitbit. These sort of devices are used to record self movement. In this project , I will try and analyze dataset from the measurements of activities by group of enthusiasts. I will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading libraries

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.2.4
```

```
## 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 Dataset

Dataset to develop model and validate model is downloaded from provided link.

The training data for this project are available here: training dataset

The test data are available here: testing dataset

To load data to R, you can download it manually or by using following commands:

```
train_file <- "pml-training.csv"
test_file <- "pml-testing.csv"
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
if (!file.exists(train_file)){
    download.file(train_url)
}

test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists(test_file)){
    download.file(test_url)
}
```

After downloading the dataset, load the dataset into R

```
train_data <- read.csv(train_file, na.strings = c("#DIV/0!","NA"))
final_test_data <- read.csv(test_file, na.strings = c("#DIV/0!","NA"))
```

Cleaning Data

First five columns(X,user_name,raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp) has no significance in building a prediction model. So, remove first five columns

```
train_data <- subset(train_data, select = -(1:5))

# remove variables with nearly zero variance
zerovarIndex <- nearZeroVar(train_data)
train_data <- train_data[, -zerovarIndex]

# remove variables that are almost always NA
mostlyNA <- sapply(train_data, function(x) mean(is.na(x))) > 0.9
train_data <- train_data[, mostlyNA == F]
```

Model Building

I decided to use RandomForest model to see if it returns acceptable performance. I will be using `train` function in `caret` package to train the model and use 10-fold cross validation.

```
#partition the dataset into train and test set
dataIndex <- createDataPartition(train_data$classe, p = 0.7, list = FALSE)
training_set <- train_data[dataIndex,]
testing_set <- train_data[-dataIndex,]
```

```
modelcontrol <- trainControl(method = "cv", number = 10, verboseIter = FALSE)
rfFit <- train(classe ~ ., method = "rf", data = training_set, trControl = modelcontrol)
```

Lets use boosting algorithm with 10-fold cross validation to predict classe.

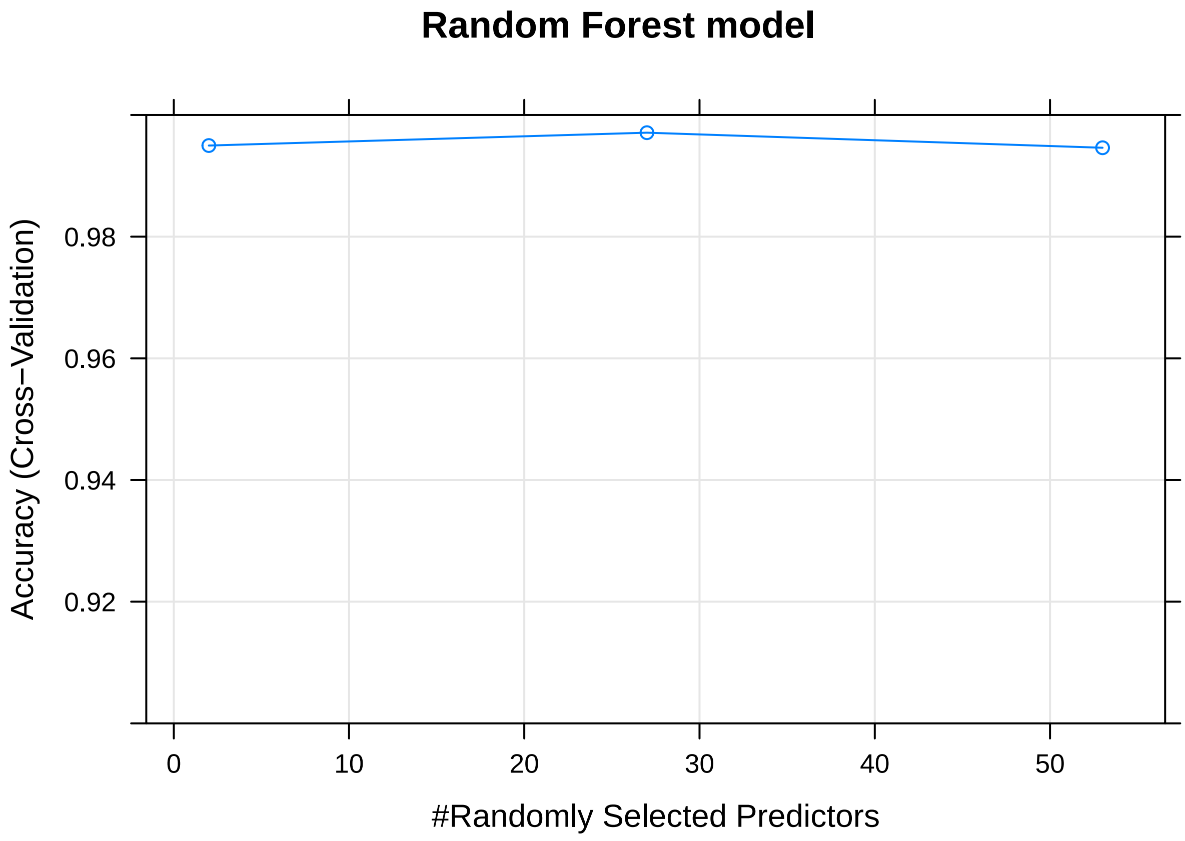
```
boostFit <- train(classe ~ ., method = "gbm", data = training_set, verbose = FALSE, trControl = modelcon
```

```
## Loading required package: plyr
```

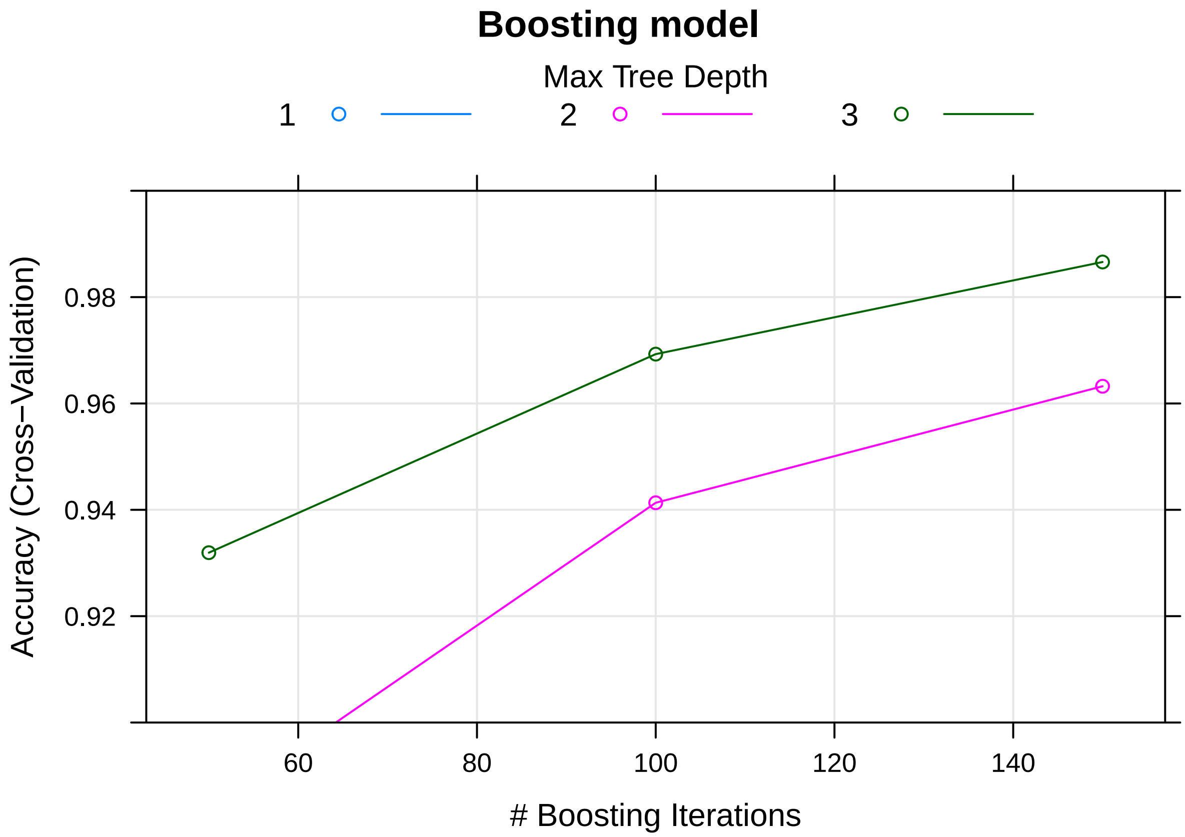
Random Forest vs Boosting Model Evaluation

Use the fitted model to predict the classe in testing dataset. Confusion matrix will compare predicted vs actual values.

```
plot(rfFit, ylim = c(0.9, 1), main = "Random Forest model")
```



```
plot(boostFit, ylim = c(0.9, 1), main = "Boosting model")
```



```
# use the random forest model fitted to predict classe in testing set
rfFit_predicted <- predict(rfFit, newdata = testing_set)

# show confusion matrix to get estimate of out-of-sample error from prediction
confusionMatrix(testing_set$classe, rfFit_predicted)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B    1 1135    2    1    0
##          C    0    2 1024    0    0
##          D    0    0    1  963    0
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9982   0.9971   0.9969   0.9991
## Specificity            0.9998   0.9992   0.9996   0.9998   0.9996
## Pos Pred Value         0.9994   0.9965   0.9981   0.9990   0.9982
## Neg Pred Value         0.9998   0.9996   0.9994   0.9994   0.9998
## Prevalence             0.2845   0.1932   0.1745   0.1641   0.1837
## Detection Rate         0.2843   0.1929   0.1740   0.1636   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9996   0.9987   0.9983   0.9983   0.9993
```

```r
# use the boosting model fitted to predict classe in testing set
boostFit_predicted <- predict(boostFit, newdata = testing_set)

# show confusion matrix to get estimate of out-of-sample error from prediction
confusionMatrix(testing_set$classe, boostFit_predicted)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    3    0    1    0
##          B   13 1115   10    1    0
##          C    0   12 1011    3    0
##          D    1    2    8  953    0
##          E    0    7    0    7 1068
##
## Overall Statistics
##
##                Accuracy : 0.9884
##                  95% CI : (0.9854, 0.991)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9854
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9917   0.9789   0.9825   0.9876   1.0000
## Specificity            0.9990   0.9949   0.9969   0.9978   0.9971
## Pos Pred Value         0.9976   0.9789   0.9854   0.9886   0.9871
## Neg Pred Value         0.9967   0.9949   0.9963   0.9976   1.0000
## Prevalence             0.2862   0.1935   0.1749   0.1640   0.1815
## Detection Rate         0.2838   0.1895   0.1718   0.1619   0.1815
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9954   0.9869   0.9897   0.9927   0.9985
```

From above comparison, random forest is the best model that can be used to fit the dataset. Out of Sample error ===

```
## Calculate OOS Error
missClass = function(values, predicted) {
        sum(predicted != values) / length(values)
}
OOS_errRateRF = missClass(testing_set$classe, rfFit_predicted)
OOS_errRateRF
```

```
## [1] 0.001699235
```

Estimated out of sample error rate for the random forests model is 0.0016992 as reported by the final model.

Final Prediction

Finally, predicting the classe of testing dataset provided using the model selected and writing the result to files.

```
# predict on test set
preds <- predict(rfFit, newdata = final_test_data)

# convert predictions to character vector
preds <- as.character(preds)

# create function to write predictions to files
pml_write_files <- function(x) {
    n <- length(x)
    for (i in 1:n) {
        filename <- paste0("predicted_output/problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE, col.names = FALSE)
    }
}

# create prediction files to submit
pml_write_files(preds)
```