Peer Graded Assignment: Prediction Assignment Writeup

henryd kasereka@@

11/27/2020

Overview

The goal of this project is to predict how they did the exercise. This is the "class" variable in the training set. You can use any of the other variables to predict with.

Background

Using devices like Jawbone Up, Nike FuelBand, and Fitbit, it is now possible to collect a large amount of personal activity data at a relatively low price. These types of devices are part of the quantified self-movement - a group of enthusiasts who regularly take action on themselves to improve their health, to find role models in their behavior, or because they are tech geeks. One thing people do on a regular basis is quantify how much of a particular activity they do, but they rarely quantify how much they do. In this project, your goal will be to use data from the accelerometers on the waistband, forearm, arm, and dumbbell of 6 participants.

Pre-processing

Packages needed to perform this project

```
library(readr)
library(e1071)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

Loading and processing

```r
train_link_data="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_link_data="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(train_link_data,destfile="training_data.csv")
download.file(test_link_data,destfile="testing_data.csv")
training_data<-read.csv("training_data.csv",na.strings = c("NA", "#DIV/0!", ""))
testing_data <- read.csv("testing_data.csv",na.strings = c("NA", "#DIV/0!", ""))
# Data dimensions
```

```r
dim(training_data)
```

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

```r
dim(training_data)
```

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

Removing columns that contains irrelevant variables and NA values

```r
training_data<- training_data[, which(colSums(is.na(training_data)) == 0)]
testing_data <- testing_data[, which(colSums(is.na(testing_data)) == 0)]
training_data <- training_data[,-c(1:7)] ##the first 7 columns are variables that has no relationship w
testing_data <- testing_data[,-c(1:7)]

dim(training_data)
```

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

```r
dim(training_data)
```

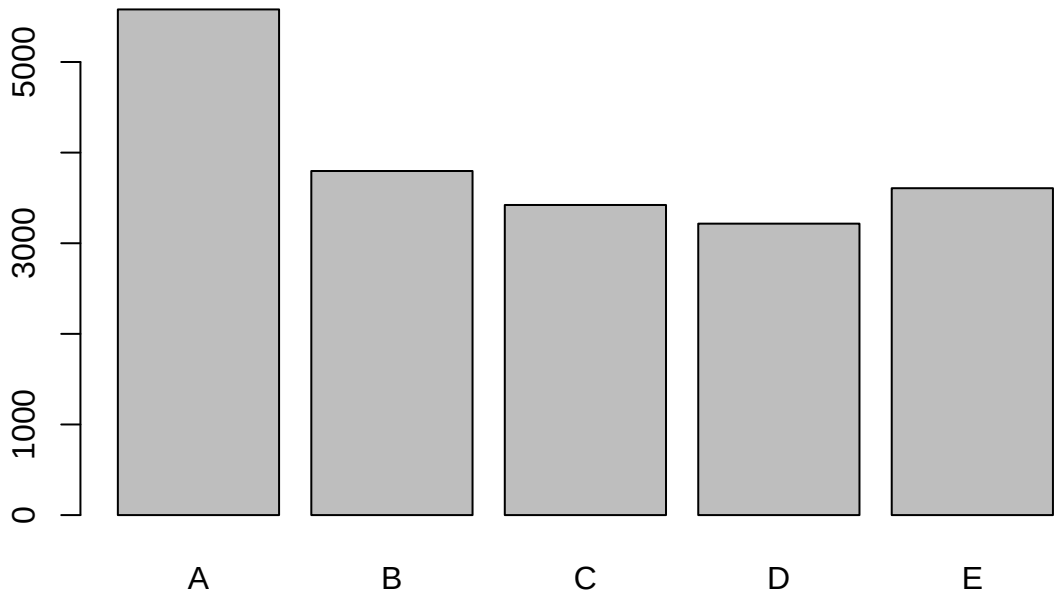
```
## [1] 19622    53
```

Checking which column names are common among testing and training.

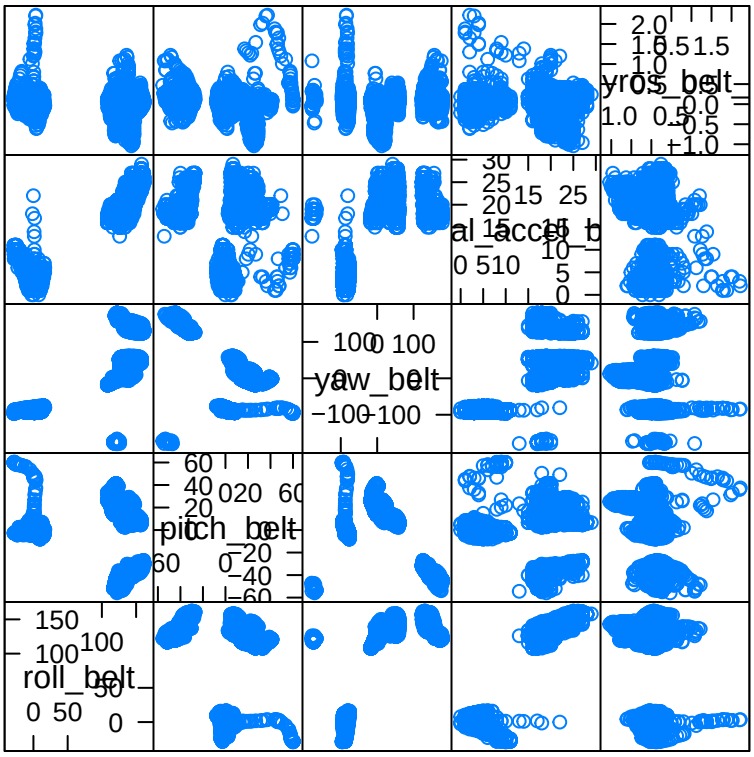
```r
length(intersect(colnames(training_data),colnames(testing_data)))
```

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

```r
barplot(table(training_data$classe))
```




```
splom(classe~training_data[1:5], data = training_data)
```



Scatter Plot Matrix

52 variables in common, everyone except class, and the target variable is fairly uniform across different classes

Partioning the training set into training and cross validation datasets

```
set.seed(123)
training_data = data.frame(training_data)
pTrain <- createDataPartition(training_data$classe, p=0.70, list=F)
train <- training_data[pTrain, ]
validation <- training_data[-pTrain, ]
```

Fitting models

Random forest Modelling "rf"

Unhappy, it takes a very long time for training, but it has a high accuracy.

```
m_fit1 <- train(classe ~ ., method="rf", data=train, verbose = TRUE, trControl = trainControl(method="c
p_val1 <- predict(m_fit1, validation)
confusionMatrix(table(validation$classe, p_val1) )
```

```
## Confusion Matrix and Statistics
##
```

```
##    p_val1
##       A    B    C    D    E
##   A 1673    1    0    0    0
##   B    6 1125    8    0    0
##   C    0    5 1018    3    0
##   D    0    0   10  954    0
##   E    0    0    4    5 1073
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9947   0.9788   0.9917   1.0000
## Specificity            0.9998   0.9971   0.9983   0.9980   0.9981
## Pos Pred Value         0.9994   0.9877   0.9922   0.9896   0.9917
## Neg Pred Value         0.9986   0.9987   0.9955   0.9984   1.0000
## Prevalence             0.2853   0.1922   0.1767   0.1635   0.1823
## Detection Rate         0.2843   0.1912   0.1730   0.1621   0.1823
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9959   0.9886   0.9948   0.9991
```

Regression tree Modelling "rpart"

```
m_fit2 <- train(classe ~ ., method="rpart", data=train)
p_val2 <- predict(m_fit2, validation)
confusionMatrix(table(validation$classe, p_val2) )
```

```
## Confusion Matrix and Statistics
##
##    p_val2
##       A    B    C    D    E
##   A 1530   28  114    0    2
##   B  464  397  278    0    0
##   C  469   30  527    0    0
##   D  440  169  355    0    0
##   E  144  145  306    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4997
##                  95% CI : (0.4869, 0.5126)
##     No Information Rate : 0.5178
```

```
##     P-Value [Acc > NIR] : 0.9973
##
##                   Kappa : 0.3464
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5021  0.51625  0.33354       NA  0.99591
## Specificity            0.9493  0.85496  0.88409   0.8362  0.88973
## Pos Pred Value         0.9140  0.34855  0.51365       NA  0.45009
## Neg Pred Value         0.6398  0.92162  0.78329       NA  0.99958
## Prevalence             0.5178  0.13067  0.26848   0.0000  0.08309
## Detection Rate         0.2600  0.06746  0.08955   0.0000  0.08275
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7257  0.68561  0.60882       NA  0.94282
```

boosted trees Modelling "gbm"

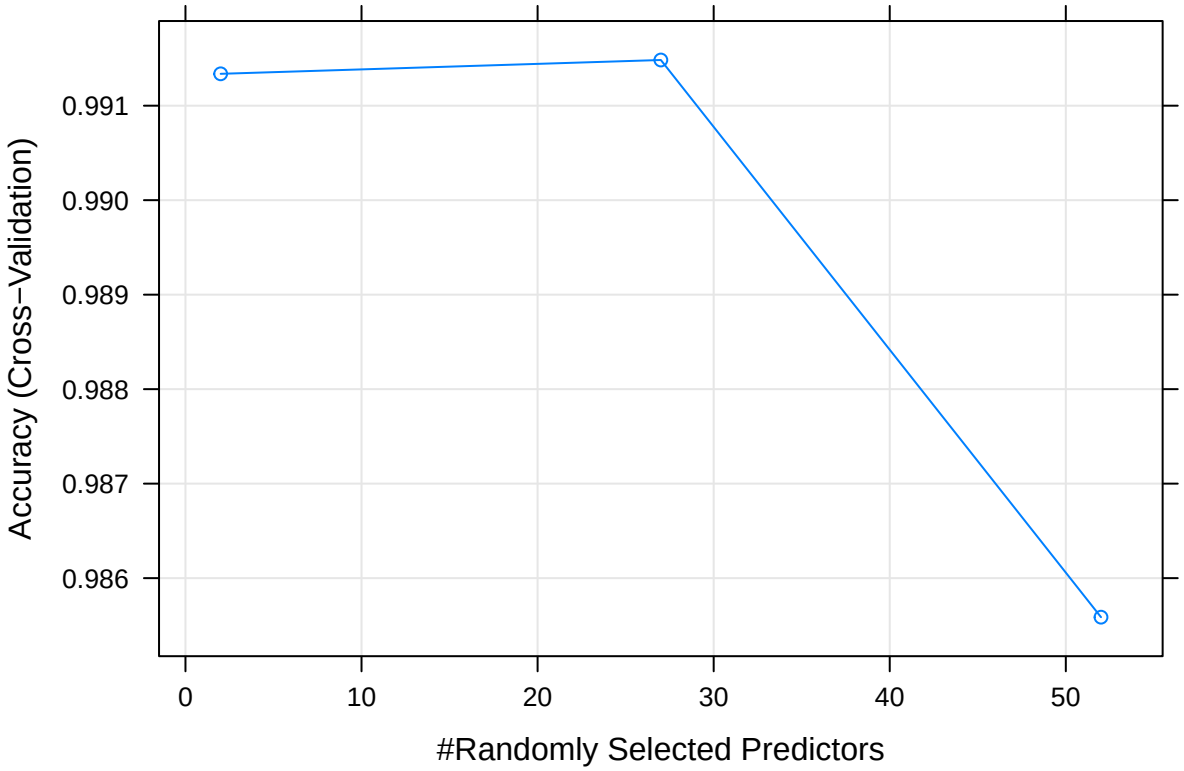
```
m_fit3 <- train(classe ~ ., method="gbm", data=train,trControl=trainControl(method = "repeatedcv", numbe
p_val3 <- predict(m_fit3, validation)
confusionMatrix(table(validation$classe, p_val3))
```

```
## Confusion Matrix and Statistics
##
##    p_val3
##        A    B    C    D    E
##   A 1650   14    5    1    4
##   B   29 1074   35    1    0
##   C    2   27  981   14    2
##   D    2    4   27  921   10
##   E    3    5   19   15 1040
##
## Overall Statistics
##
##                Accuracy : 0.9628
##                  95% CI : (0.9576, 0.9675)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9529
##
##  Mcnemar's Test P-Value : 0.0002067
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9786   0.9555   0.9194   0.9674   0.9848
## Specificity            0.9943   0.9863   0.9907   0.9913   0.9913
## Pos Pred Value         0.9857   0.9429   0.9561   0.9554   0.9612
## Neg Pred Value         0.9915   0.9895   0.9823   0.9937   0.9967
```

```
## Prevalence             0.2865   0.1910   0.1813   0.1618   0.1794
## Detection Rate         0.2804   0.1825   0.1667   0.1565   0.1767
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9865   0.9709   0.9550   0.9794   0.9881
```

Comparing the three modeling used above, the result shows that the random forest model has the highest precision in cross-validation. Consequently, we will use the random forest model to predict the test samples.

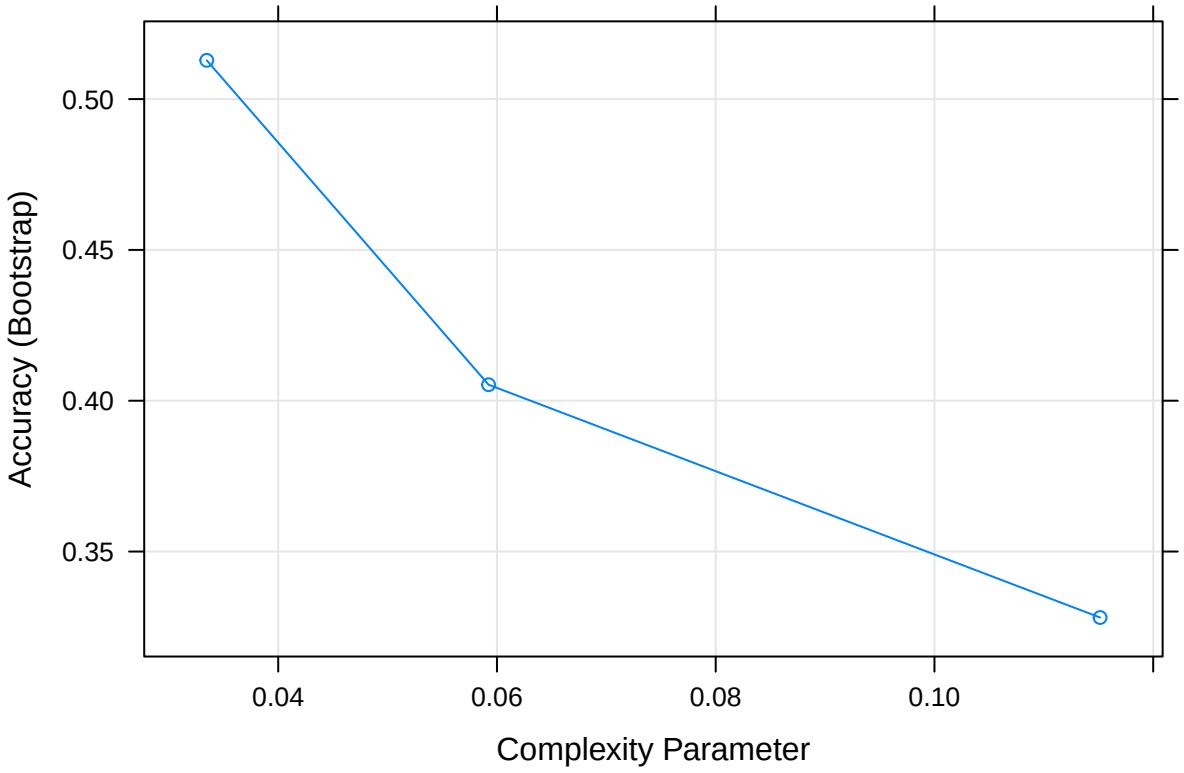
```
plot(m_fit1)
```




```
par(new=TRUE)
```

```
## Warning in par(new = TRUE): calling par(new=TRUE) with no plot
```

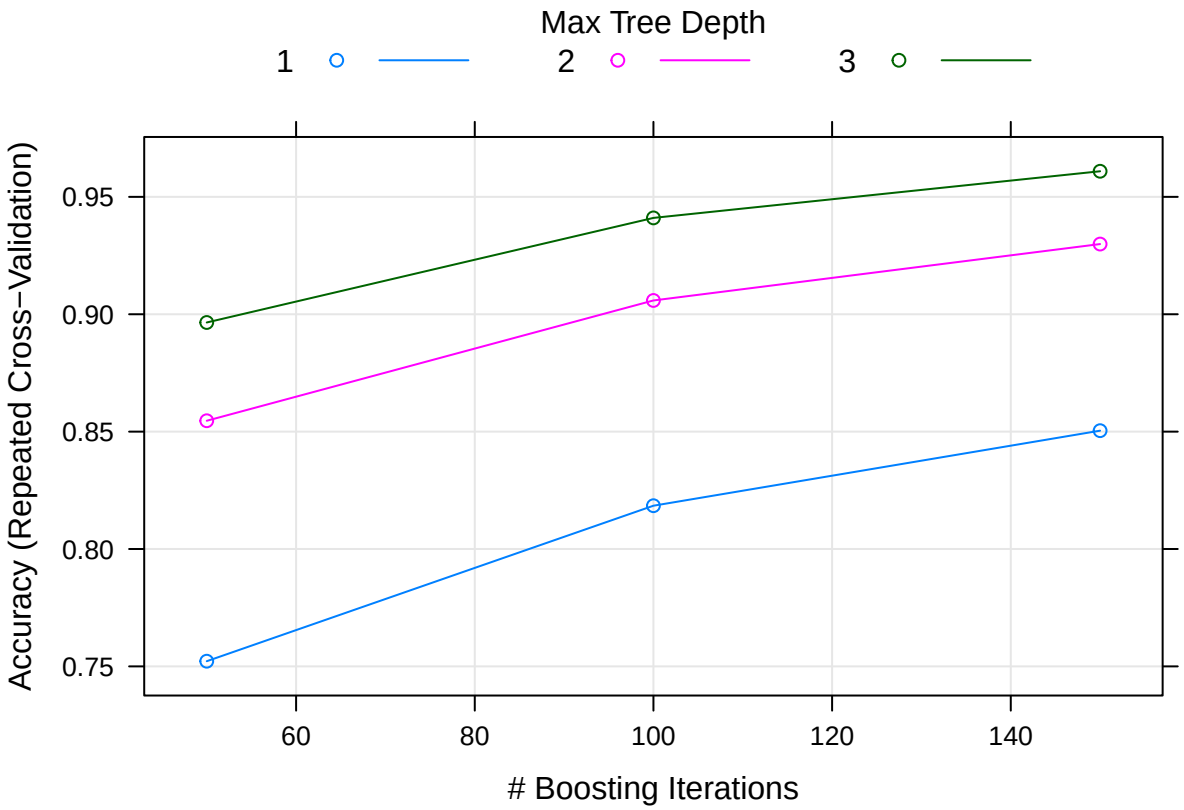
```
plot(m_fit2)
```




```
par(new=TRUE)
```

```
## Warning in par(new = TRUE): calling par(new=TRUE) with no plot
```

```
plot(m_fit3)
```



Prediction

According to the result obtained above, we choose to use a random forest model for the prediction

```
testing_data <- testing_data[, colSums(is.na(testing_data)) == 0]
testing_data <- testing_data[, -(1:5)]
near_zvt <- nearZeroVar(testing_data)
testing_data <- testing_data[, -near_zvt]
```

```
pred_f <- predict(m_fit1, data = testing_data)
head(pred_f)
```

```
## [1] A A A A A A
## Levels: A B C D E
```