Practical Machine Learning Week 4 Assignment

Moksh Goyal

September 24, 2016

Background of the Project

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, the 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.

Loading the Data sets.

```
train <- read.csv("C:/Users/Sony/Documents/pml-training.csv")
test <- read.csv("C:/Users/Sony/Documents/pml-testing.csv")
dim(train)
```

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

```
dim(test)
```

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

From here we could see that there are 19622 rows in training set and 160 columns in training set. Also there are 20 in testing set and 160 columns in testing set.

Loading the libraries which will be used in the Program

```
library(caret)
```

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

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

```
library(caTools)
library(rpart)
library(rpart.plot)
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(rattle)

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Removing irrelevant variables which are not going to have any impact on the category. Also removing classe variable and problem id from the train and test set to combine the data sets later for cleaning

```
train1 <- select(train,select = -c(classe,X,user_name , raw_timestamp_part_1
, raw_timestamp_part_2 , cvtd_timestamp , new_window,num_window))
test1 <- select(test,select = -c(problem_id , X , user_name ,
raw_timestamp_part_1 , raw_timestamp_part_2 , cvtd_timestamp ,
new_window,num_window))
```

Combining the train and test set and then removing the variables which have even a single NA value.

```
comb <- rbind(train1,test1)
comb <- comb[,colSums(is.na(comb)) == 0]
```

Number of columns left after removing the irrelevant and NA variable is 52 .

Checking for variables with low variance. These variables will have least impact on deciding the category.

```
lowvar <- nearZeroVar(comb,saveMetrics = TRUE)
comb <- comb[,lowvar[,'nzv']== 0]
```

From checking the dimensions of comb after conducting the variance exercise we came to know that no variable was removed due to low variance attribute.

Finding variables which are highly correlated. We would like to remove the variables which are highly correlated.

```
tmp <- cor(comb)
tmp[upper.tri(tmp)] <- 0
diag(tmp) <- 0
comb <- comb[,!apply(tmp,2,function(x) any(x > 0.9))]
```

Here, we have decided to keep the threshold value of 0.9 to remove the highly correlated variables. After removing the highly correlated variables we are left with 49 columns

Separating the combined data set into training and testing set

```
train1 <- comb[1:19622,]
test1 <- comb[19623:19642,]
```

Adding the removed variables from training and testing set

```
train1$classe <- train$classe
id <- 1:20
test1$id <- id
```

Dividing the training set for training and cross validation

```
set.seed(33)
spl <- sample.split(train1,SplitRatio = 0.7)
train2 <- subset(train1,spl == TRUE)
cv <- subset(train1,spl == FALSE)
```

Here we have split the training data set into training set and cross validation set in the ratio of 70% - 30%.

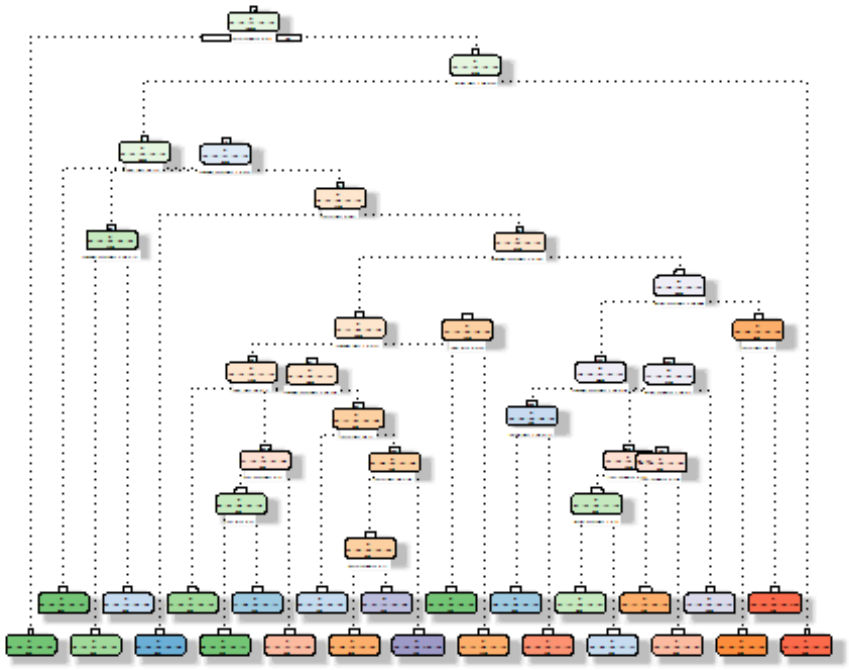
Building a Regression Tree on the training set

```
set.seed(23)
model1 <- rpart(classe~.,data = train2,method = "class",minbucket = 50)
pred <- predict(model1,newdata = cv,type = "class")
```

Building a fancy tree

```
fancyRpartPlot(model1)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Making the predictions on the cross validation set

```
table1 <- table(cv$classe,pred)
table1
```

```
##    pred
##        A    B    C    D    E
##   A 1471   67    4  115   17
##   B  151  646  115  141   86
##   C   30   66  822   42   66
##   D   58   67  141  619   79
##   E   89  146   92  119  637
```

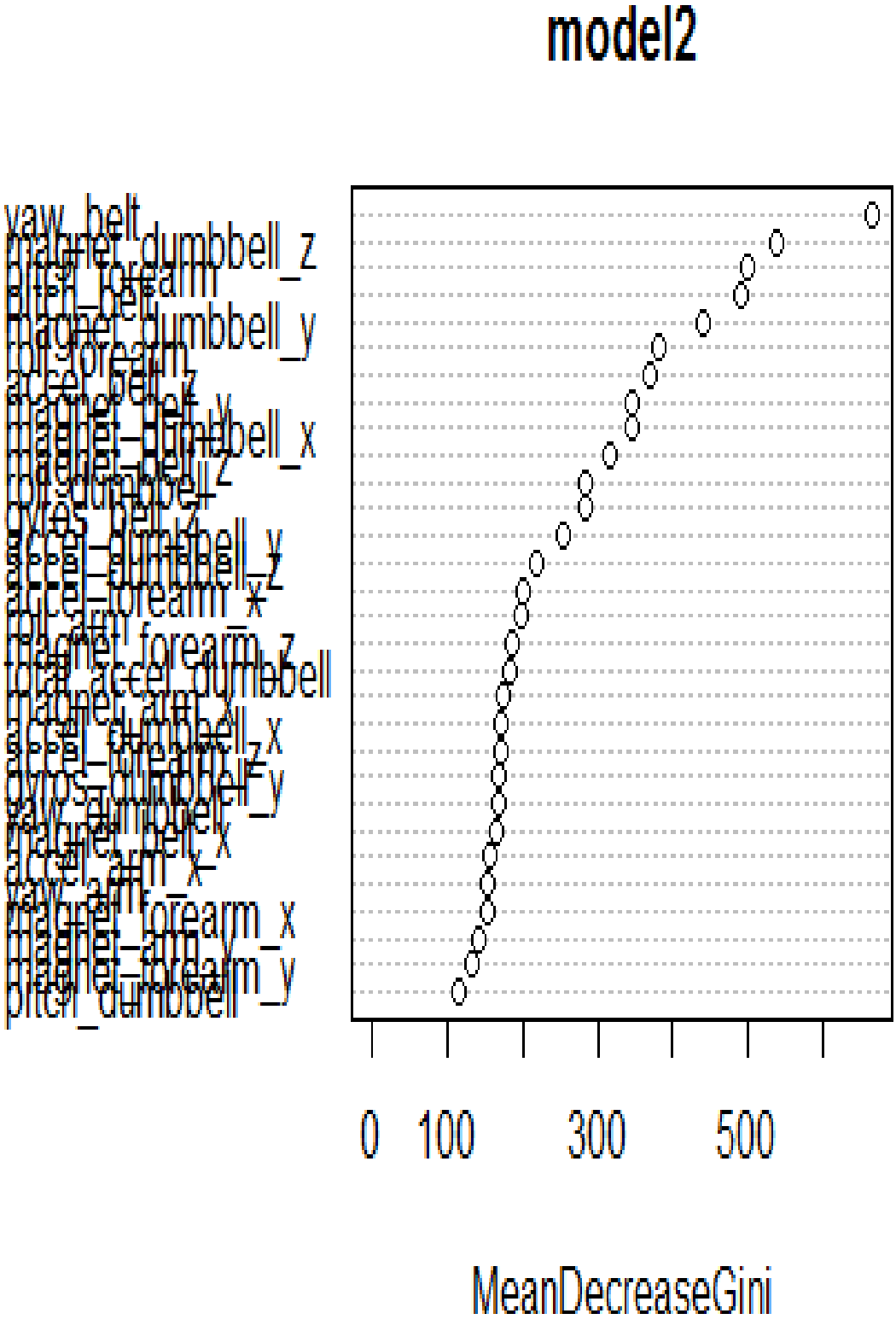
From here we could see that the accuracy of the regression tree is 0.7127081 which is pretty less. So we will build a random Forest to improve our accuracy.

Building a Random Forest

```
set.seed(23)
model2 <- randomForest(classe~.,data = train2,nodesize = 25,ntree = 200)
pred2 <- predict(model2,newdata = cv)
```

To check the variable importance in Random Forest

```
varImpPlot(model2)
```


model2
yaw_belt
magnet_dumbbell_z
magnet_dumbbell_y
magnet_dumbbell_x
accel_dumbbell_z
accel_dumbbell_y
accel_forearm_x
roll_forearm
forearm_z
accel_dumbbell
magnet_arm_z
gyros_dumbbell_x
gyros_dumbbell_y
pitch_forearm
magnet_forearm_x
magnet_arm_y
magnet_forearm_y
pitch_dumbbell
0
100
300
500
MeanDecreaseGini

Making the predictions on the cross validation set

```
table2 <- table(cv$classe,pred2)
table2
```

```
##    pred2
##        A    B    C    D    E
##   A 1669    4    0    0    1
##   B   10 1126    3    0    0
##   C    0   12 1012    2    0
##   D    0    0   23  939    2
##   E    0    0    1    9 1073
```

From here we could see that the accuracy of the Random Forest is 0.9886171 which is pretty awesome.

Making final predictions for the testing set.

```
pred3 <- predict(model2,newdata = test1)
pred3
```

```
## 19623 19624 19625 19626 19627 19628 19629 19630 19631 19632 19633 19634
##     B     A     B     A     A     E     D     B     A     A     B     C
## 19635 19636 19637 19638 19639 19640 19641 19642
##     B     A     E     E     A     B     B     B
## Levels: A B C D E
```