# Manner Prediction

**Sameul Wang**

2019-04-15

## 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. More information is available from the website (see the section on the Weight Lifting Exercise Dataset).

**Data**

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

## Assignment

The goal of your 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.

## My_Solution

From the website discription, we know this data set is structured as:

"Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E)

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz5kwIVninr"

```r
url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(url1,"training.csv",method="curl")
download.file(url2,"testing.csv",method="curl")
```

```
training<-read.csv(file = "training.csv",sep = ",",header = TRUE)
testing<-read.csv(file = "testing.csv",sep = ",",header = TRUE)

dim(training)
```

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

```
dim(testing)
```

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

```
summary(training$classe)
```

```
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

By a glimps of the data, these are sevral column contains NA value, and some column containing useless information to our model. Thus we will delete those side-effect column. Also, those near zero columns should be dropped from model.

```
library(caret)
training<- training[,-c(1,3:7)]
training<-training[,!apply(is.na(training),2,sum)]
training<-training[,-nearZeroVar(training)]
names(training)
```

```
##  [1] "user_name"            "roll_belt"            "pitch_belt"          
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"        
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"        
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"       
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"            
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"     
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"         
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"         
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"        
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"        
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"    
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"    
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"   
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"       
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"     
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"     
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"    
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

Seeing from the data, our goal is clear to build the fit model from and to validate in **training dataset**. Then use this model to predict 20 values in the **testing dataset**.

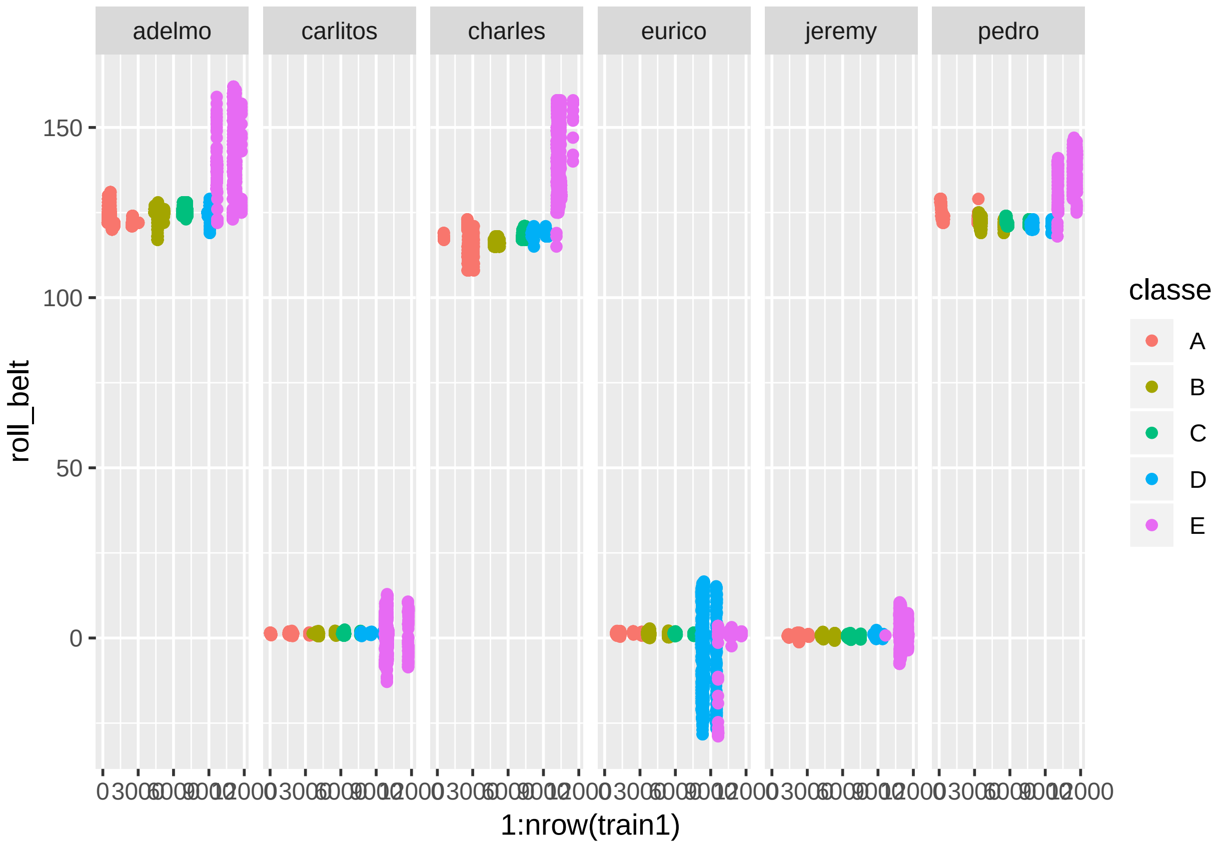
The error rate of our model will be a comparison between prediction value and real results in **training dataset**.

As training set is quite large with an observation of 19622, we will split it into 2 groups, 40% for testing, 60% for training.

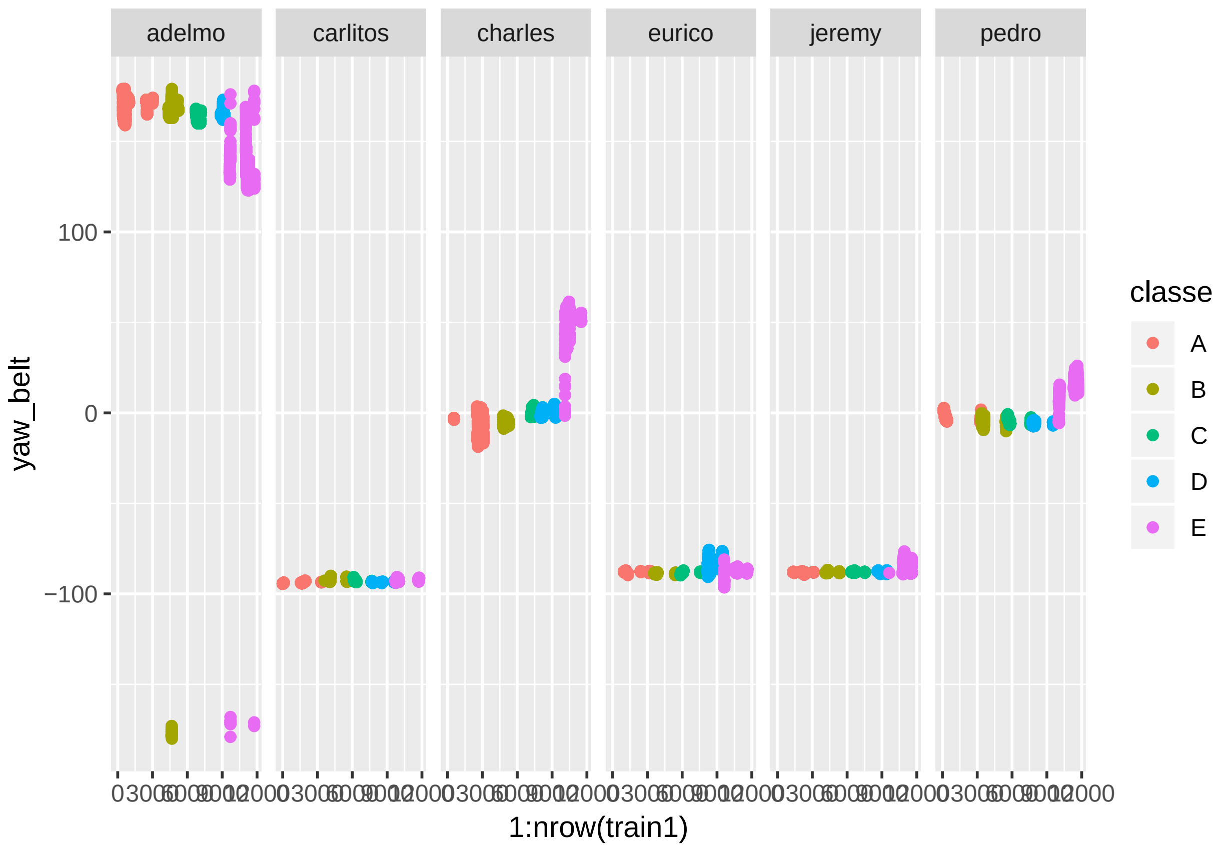
```
inTrain<-createDataPartition(y = training$classe,p = 0.6,list=FALSE )
test<-training[-inTrain,]
train<-training[inTrain,]
```

Then we look at the 'train' data to pickup some features for model. We know there are 6 participants, 4 sensors (with several measurements for each), and 5 activities. Let's generally look at 5 activities performancfe by 6 participants.

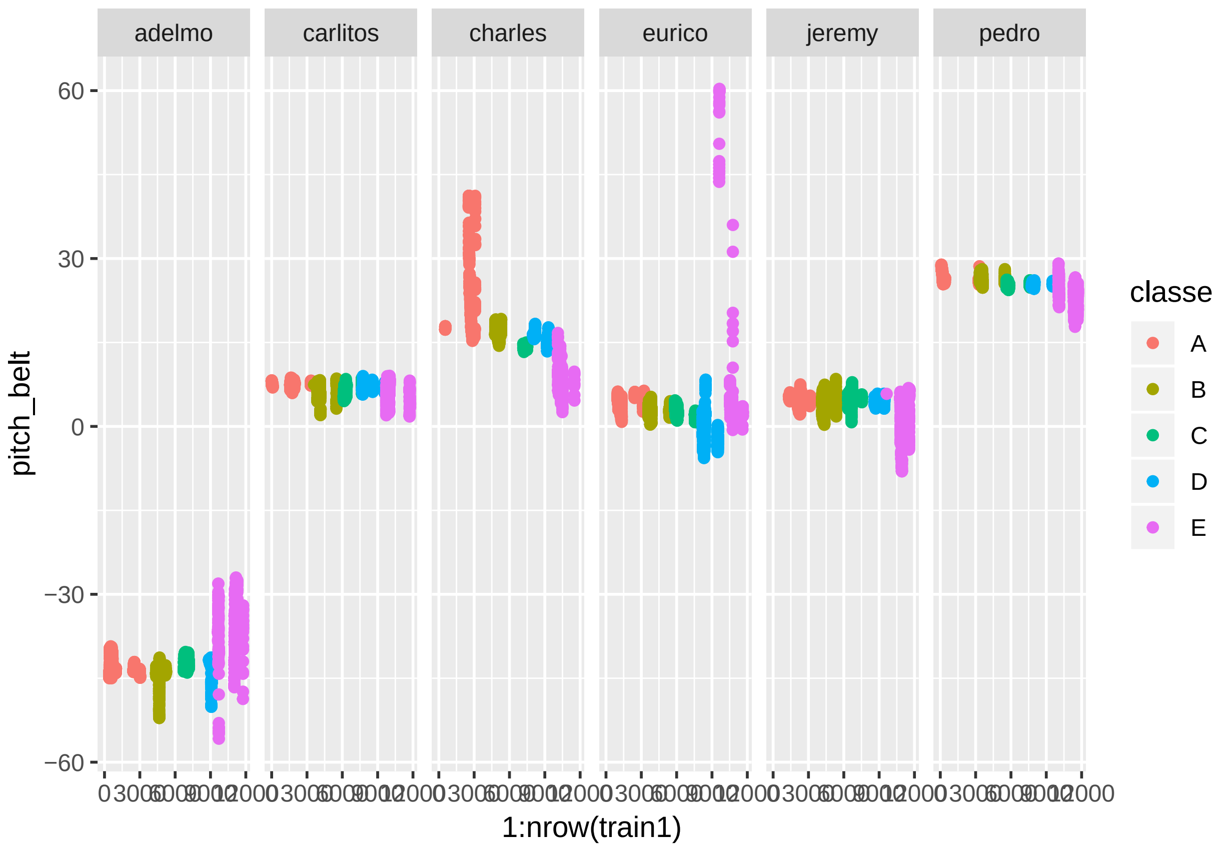
```
library(ggplot2)
train1<-train[,c("classe","user_name","roll_belt","pitch_belt","yaw_belt")]

g1<-ggplot(data = train1, mapping = aes(x=1:nrow(train1),y = roll_belt,color=classe))+facet_grid(.~user_
g1
```



```
g2<-ggplot(data = train1, mapping = aes(x=1:nrow(train1),y = yaw_belt,color=classe))+facet_grid(.~user_
g2
```



```
g3<-ggplot(data = train1, mapping = aes(x=1:nrow(train1),y = pitch_belt,color=classe))+facet_grid(.~use
g3
```



The truth is we can not distingush some obvious features that can directly discriminate calsse, so that we can not fit a model with the physical and interpretable way. Therefore we will try the statistical way with the expense of lowering down interpret ability.

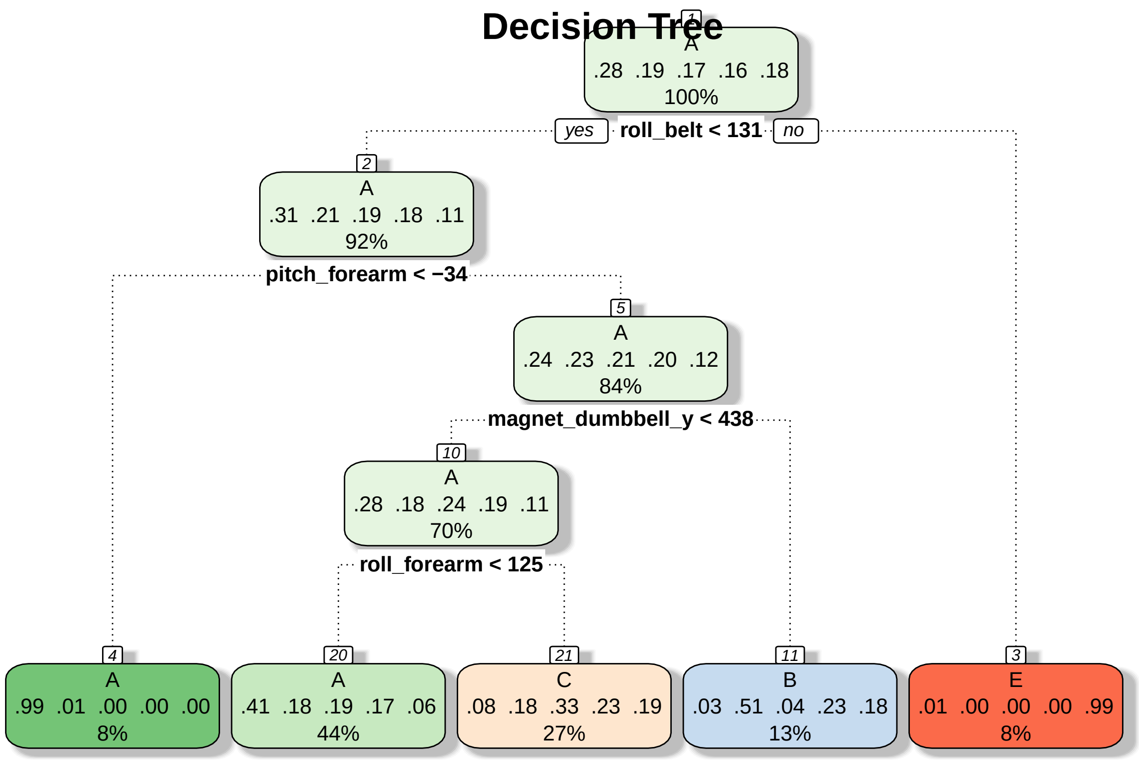
First, we will try relatively higher interpretability method, decision tree, to fit a model. The decison tree is a way to split data into 2 sub groups with the goal of best homogeneity on each leaves. To prevent overfit, we will prune to return best tree model.

```r
library(rattle)
library(rpart)
mdl1<-train(classe~.,data=train, method= "rpart")
mdl1
```

```
## CART
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03618889  0.5014974  0.35215623
##   0.06063123  0.4116552  0.20269944
##   0.11438064  0.3256131  0.06435399
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03618889.
```

```
pfit<- prune(mdl1$finalModel, cp=mdl1$finalModel$cptable[which.min(mdl1$finalModel$cptable[,3]),"CP"])
fancyRpartPlot(pfit,main = "Decision Tree", sub="")
```



We can see, a simple decision tree model is failed to reduce internal variance, which returns with low accurary.

Then We will try the k-folds cross-validation plus random forest to help pick up the best predictors. The k-folds cross validation is a method to split data set into k random uniform groups. One by one, each group will be chosed as the testing while others as training. Finally choose the one with minimun out-of-model error rate to acheive maximun generalizaion. The random forest is this tech to bootstrap and average samples on each leaves of the desicion tree to achieve highly discrimination accuracy with the cost of interpret ability and precessing time.

```
set.seed(123)
mdl2<-train(classe~.,data=train, method= "rf",
            trControl=trainControl(method="CV", number=10),verboseIter= TRUE)
mdl2
```

```
## Random Forest
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10598, 10597, 10600, 10599, 10597, 10599, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9890444  0.9861384
##   29    0.9895537  0.9867850
##   57    0.9819092  0.9771120
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 29.
```

Application of random forest is effective.

```
confusionMatrix(predict(mdl2, test),test$classe)
```

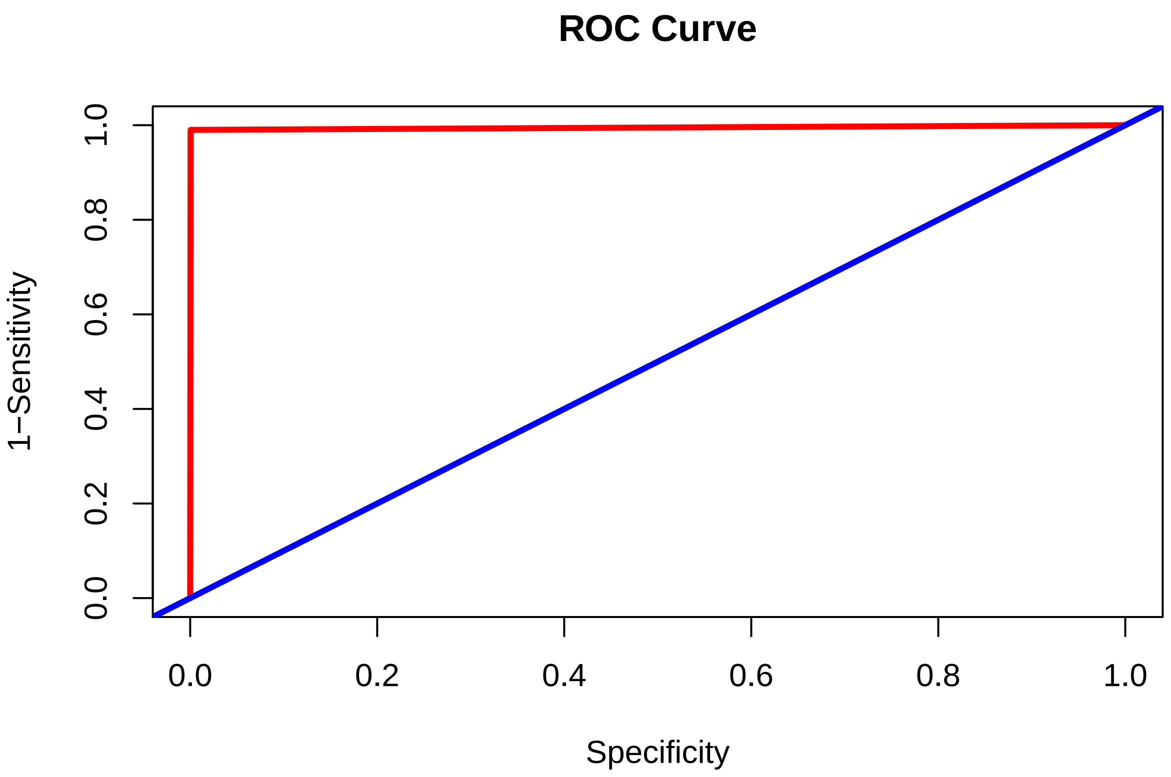
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   15    0    0    0
##          B    1 1501    5    0    0
##          C    0    2 1352   21    1
##          D    0    0   11 1265    4
##          E    0    0    0    0 1437
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9902, 0.9942)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9888   0.9883   0.9837   0.9965
## Specificity            0.9973   0.9991   0.9963   0.9977   1.0000
## Pos Pred Value         0.9933   0.9960   0.9826   0.9883   1.0000
## Neg Pred Value         0.9998   0.9973   0.9975   0.9968   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1913   0.1723   0.1612   0.1832
## Detection Prevalence   0.2863   0.1921   0.1754   0.1631   0.1832
## Balanced Accuracy      0.9984   0.9939   0.9923   0.9907   0.9983
```

Generalize this model cunstrusted from 60% data to the remaining 40% test data. The out of sample error is acceptable with **accurary 0.9915 (that is a 0.0085 out of sample rate)** and Kappa 0.9892.

```
library(pROC)
roc<-roc(response = test$classe,predictor = factor(predict(mdl2, test),order=TRUE))
roc
```

```
##
## Call:
## roc.default(response = test$classe, predictor = factor(predict(mdl2,     test), order = TRUE))
##
## Data: factor(predict(mdl2, test), order = TRUE) in 2232 controls (test$classe A) < 1518 cases (test$
## Area under the curve: 0.9948
```

```r
y<-roc$sensitivities
x<-1-roc$specificities

par(new=TRUE)
plot(y~x,type="l", col="red", xlab ="Specificity", ylab="1-Sensitivity",lwd=3)
abline(a = 0,b=1, col="blue",lwd=3)
title(main="ROC Curve")
```



The ROC curve proves good classification of this model with 0.9959 under curve area.