Pratical Machine Learning Final Project - Prediction Assignment Writeup

Zhenkun Guo

April 24, 2016

1. Project Introduction

1.1. Background

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 here. (see the section on the Weight Lifting Exercise Dataset).

1.2. 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.

1.3. Goal

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.

2. Preparation

2.1. Load Necessary Packages

Load the necessary packages ggplot2 and caret for machine learning.

```
library(ggplot2)
library(caret)
```

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

2.2. Download and Load the Data

Download the data if they do not exist and then load in.

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists("training.csv"))
        download.file(trainUrl, "training.csv", method="auto")
if (!file.exists("testing.csv"))
        download.file(testUrl, "testing.csv", method="auto")
training<-read.csv("training.csv")
testing<-read.csv("testing.csv")
dim(training)
```

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

```
dim(testing)
```

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

2.3 Set Ramdon Seed

```
set.seed(9527)
```

3. Data Cleaning

3.1. Exclude Columns with No Variance

Some measured variables has no variability, these variables will not help in the machine learning.

```
trainingNZV <- nearZeroVar(training, saveMetrics=TRUE)
training<-training[,!trainingNZV$nzv]
dim(training)
```

```
## [1] 19622   100
```

100 Variables left.

3.2. Exclude Columns with Too Many NAs

Some columns include too many NAs, these columns will not help in the machine learning. So we need to exclude these variables. We set a thredhold, if portion of NA larger than 0.8, the variable will be excluded.

```
nanum<-apply(is.na(training),2,sum)
training<-training[,!(nanum/dim(training)>0.8)]
dim(training)
```

```
## [1] 19622    59
```

59 variables left.

3.3. Exclude the ID of the Observations

ID of the observations are not relevant to the question. Also, we will make sure the classe varibale, which is our goal is a factor variable.

```
training<-training[,-1]
training$classe<-as.factor(training$classe)
dim(training)
```

```
## [1] 19622    58
```

58 Varibales left.

3.4. Only Leave the Same Set in Testing

To applied the obtained prediction method, we need to make sure the testing set has the same series of varibles as the training set.

```
colindex<-NULL
for (i in 1:length(colnames(training))){
        temp<-grep(colnames(training)[i],colnames(testing))
        if(length(temp)>0)
                colindex[i]<-temp
}
testing<-testing[,colindex]
dim(testing)
```

```
## [1] 20 57
```

The testing set has 57 variables, it does not have "classe".

3.5. Divide Training Set to Training and Testing Sets

Because we need to evaluate the quality of learning algorithms, we need testing set and here we divide the large training set to two parts.

```
inTrain<-createDataPartition(training$classe, p = 0.65, list = FALSE)
Mytraining<-training[inTrain,]
Mytesting<-training[-inTrain,]
dim(Mytraining)
```

```
## [1] 12757    58
```

```
dim(Mytesting)
```

```
## [1] 6865   58
```

As we can see, the training set has 12757 observations and the testing set has 6865 observations.

4. Apply Machine Learning Algorithm

4.1. Linear Discriminant Analysis

4.1.1. Apply the Algorithm

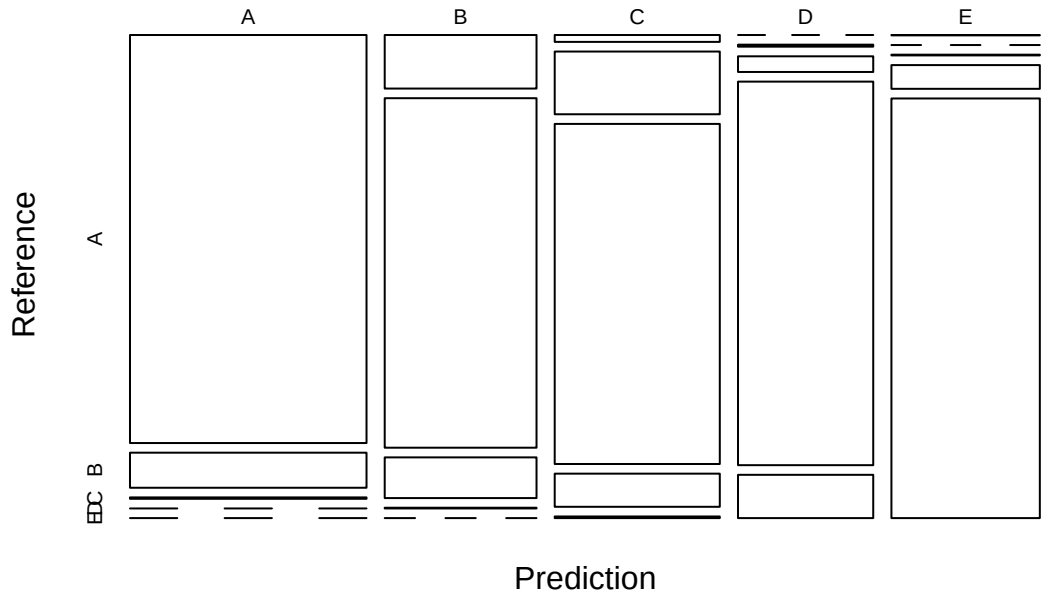
```
mod_lda<-train(classe ~ . , method = "lda" , data = Mytraining)
pred_lda<-predict(mod_lda, newdata = Mytesting)
con_lda<-confusionMatrix(pred_lda, Mytesting$classe)
```

4.1.2. Evaluate the Method

```
con_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1781  153    6    0    0
##          B  150  979  114    2    0
##          C   21  191 1036  101    5
##          D    0    5   39  957  108
##          E    1    0    2   65 1149
##
## Overall Statistics
##
##                Accuracy : 0.8597
##                  95% CI : (0.8513, 0.8679)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8227
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9119   0.7372   0.8655   0.8507   0.9105
## Specificity            0.9676   0.9520   0.9439   0.9735   0.9879
## Pos Pred Value         0.9180   0.7863   0.7651   0.8629   0.9441
## Neg Pred Value         0.9651   0.9379   0.9708   0.9708   0.9800
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2594   0.1426   0.1509   0.1394   0.1674
## Detection Prevalence   0.2826   0.1814   0.1972   0.1615   0.1773
## Balanced Accuracy      0.9398   0.8446   0.9047   0.9121   0.9492
```

```
plot(con_lda$table,col = "white", main = paste("Linear Discriminant Analysis Confusion Matrix: Accuracy
```



The accuracy is 85.97%, which is relatively low and we can see numbers of incorrect classification in all five classes.

4.2. Random Forest Algorithm

4.2.1. Apply the Algorithm

```
mod_rf<-train(classe ~ . , method = "rf" , data = Mytraining)
pred_rf<-predict(mod_rf, newdata = Mytesting)
con_rf<-confusionMatrix(pred_rf, Mytesting$classe)
```

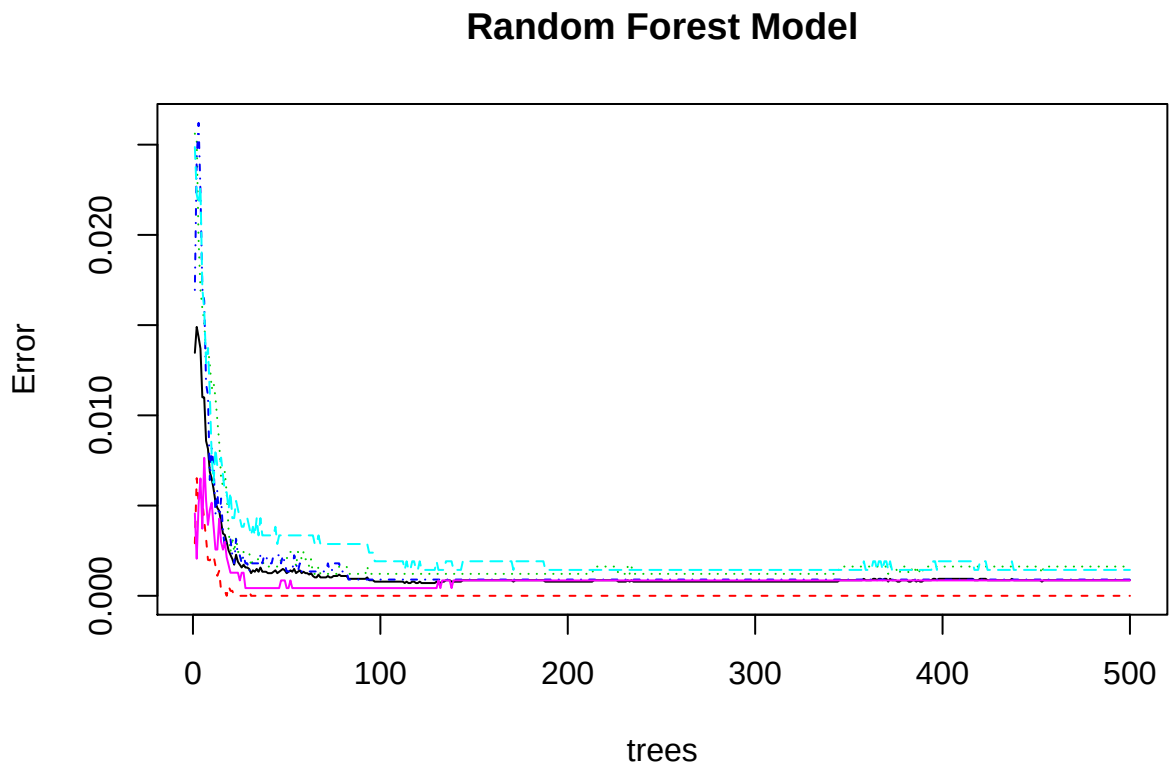
4.2.2. Evaluate the Method

```
con_rf
```

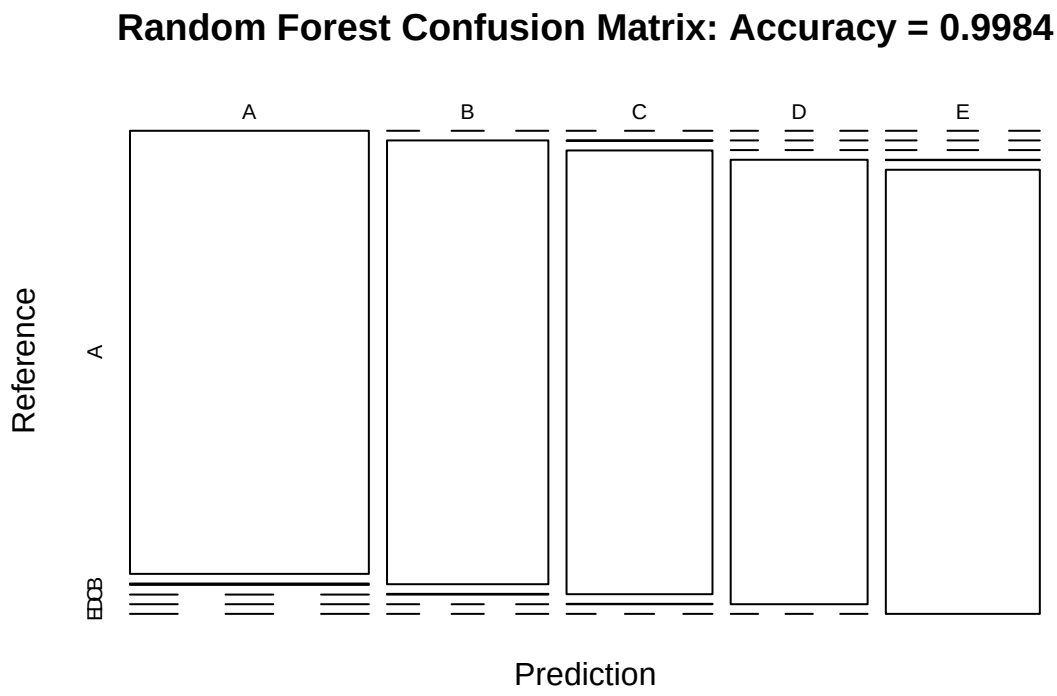
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1953    6    0    0    0
##          B    0 1321    2    0    0
##          C    0    1 1195    1    0
##          D    0    0    0 1123    0
##          E    0    0    0    1 1262
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9984
##                  95% CI : (0.9971, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.998
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9983   0.9982   1.0000
## Specificity            0.9988   0.9996   0.9996   1.0000   0.9998
## Pos Pred Value         0.9969   0.9985   0.9983   1.0000   0.9992
## Neg Pred Value         1.0000   0.9987   0.9996   0.9997   1.0000
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1924   0.1741   0.1636   0.1838
## Detection Prevalence   0.2854   0.1927   0.1744   0.1636   0.1840
## Balanced Accuracy      0.9994   0.9972   0.9990   0.9991   0.9999
```

```r
plot(mod_rf$finalModel, main = "Random Forest Model")
```



```
plot(con_rf$table,col = "white", main = paste("Random Forest Confusion Matrix: Accuracy =", round(con_r
```



We can see as the number of tree increases, the error goes down dramatically. And the accuracy is relatively high: 99.83%

4.3. Boosting Algorithm

4.3.1. Apply the Algorithm

```
mod_gbm<-train(classe ~ . , method = "gbm" , data = Mytraining)
pred_gbm<-predict(mod_gbm, newdata = Mytesting)
con_gbm<-confusionMatrix(pred_gbm, Mytesting$classe)
```

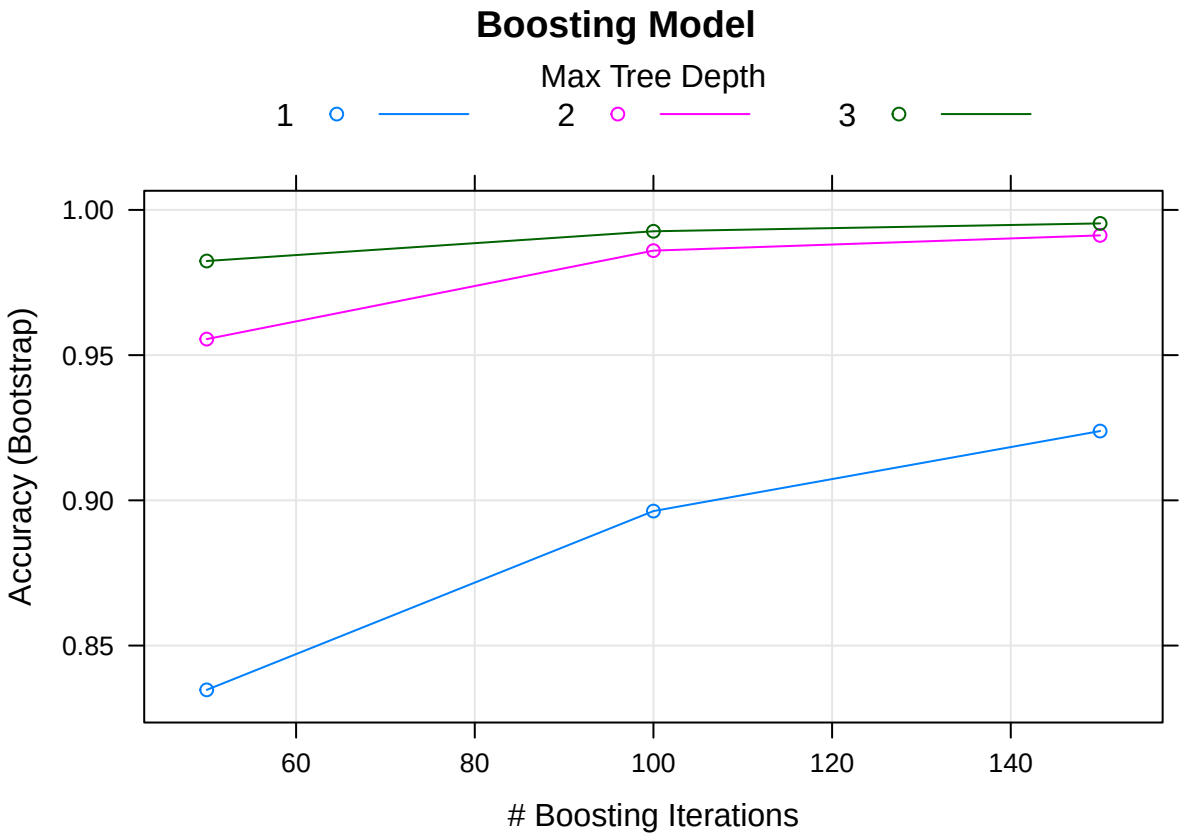
4.3.2. Evaluate the Method

```
con_gbm
```

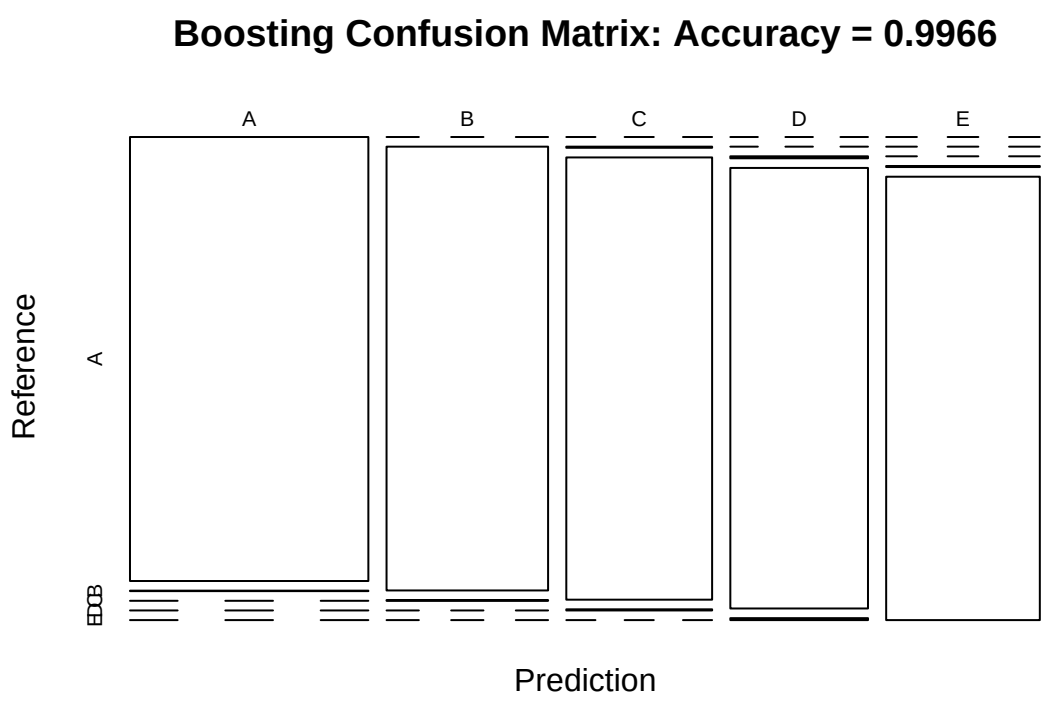
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1953    2    0    0    0
##          B    0 1323    2    0    0
```

```
##          C    0    3 1190    3    0
##          D    0    0    5 1119    5
##          E    0    0    0    3 1257
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9958
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9962   0.9942   0.9947   0.9960
## Specificity            0.9996   0.9996   0.9989   0.9983   0.9995
## Pos Pred Value         0.9990   0.9985   0.9950   0.9911   0.9976
## Neg Pred Value         1.0000   0.9991   0.9988   0.9990   0.9991
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1733   0.1630   0.1831
## Detection Prevalence   0.2848   0.1930   0.1742   0.1645   0.1835
## Balanced Accuracy      0.9998   0.9979   0.9965   0.9965   0.9978
```

```
plot(mod_gbm, main = "Boosting Model")
```




```
plot(con_gbm$table,col = "white", main = paste("Boosting Confusion Matrix: Accuracy =", round(con_gbm$o
```



We can see as the number of iterations increases, the Accuracy goes up rapidly. And the accuracy is relatively high: 99.61%

4.4. Combine Different Models

As have see here, the random forest and boosting methods did a great job. However, We'd like to see if we can have even better results by combining all the three methods.

4.4.1. Combine Models

```
Mycomb_pred<-data.frame(pred_rf, pred_gbm, pred_lda, classe = Mytesting$classe)
mod_comb<-train(classe ~ . , method = "gbm", data = Mycomb_pred)
pred_comb<-predict(mod_comb, newdata = Mytesting)
con_comb<-confusionMatrix(pred_comb, Mytesting$classe)
```

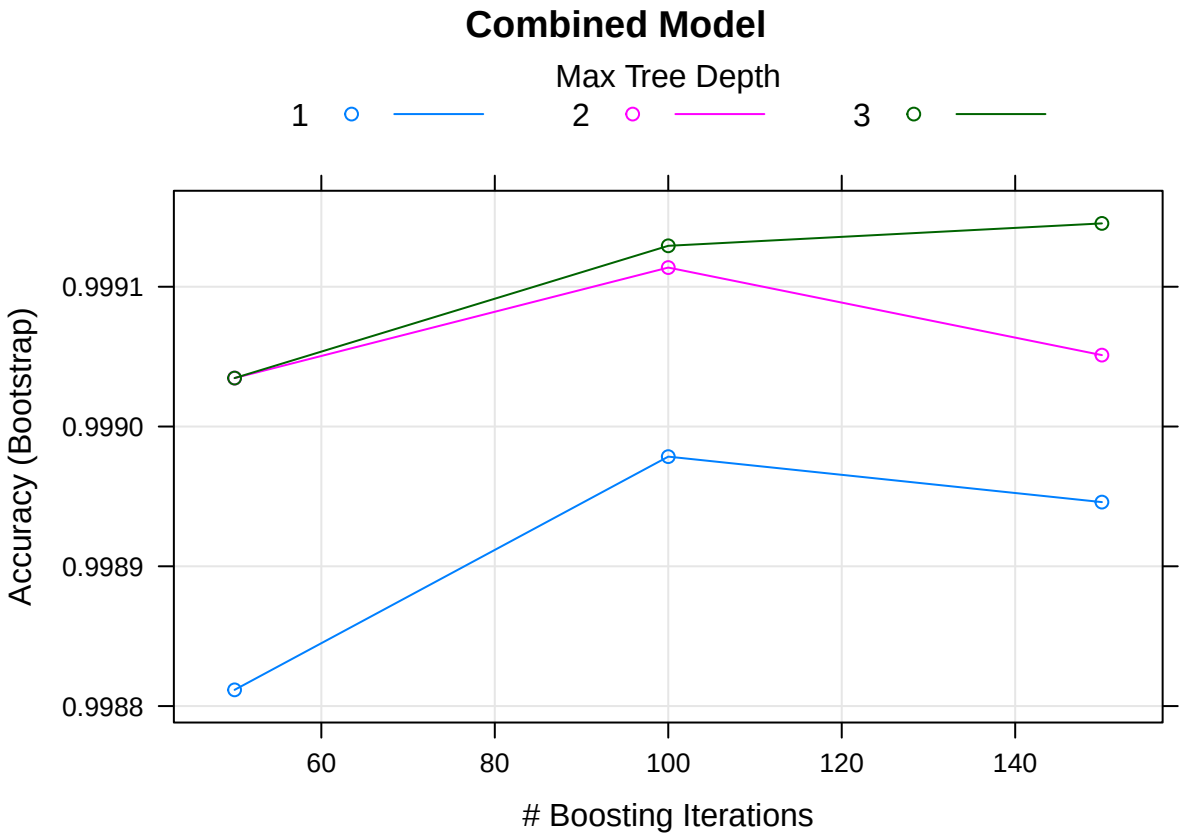
4.4.2. Evaluate the Method

```
con_comb
```

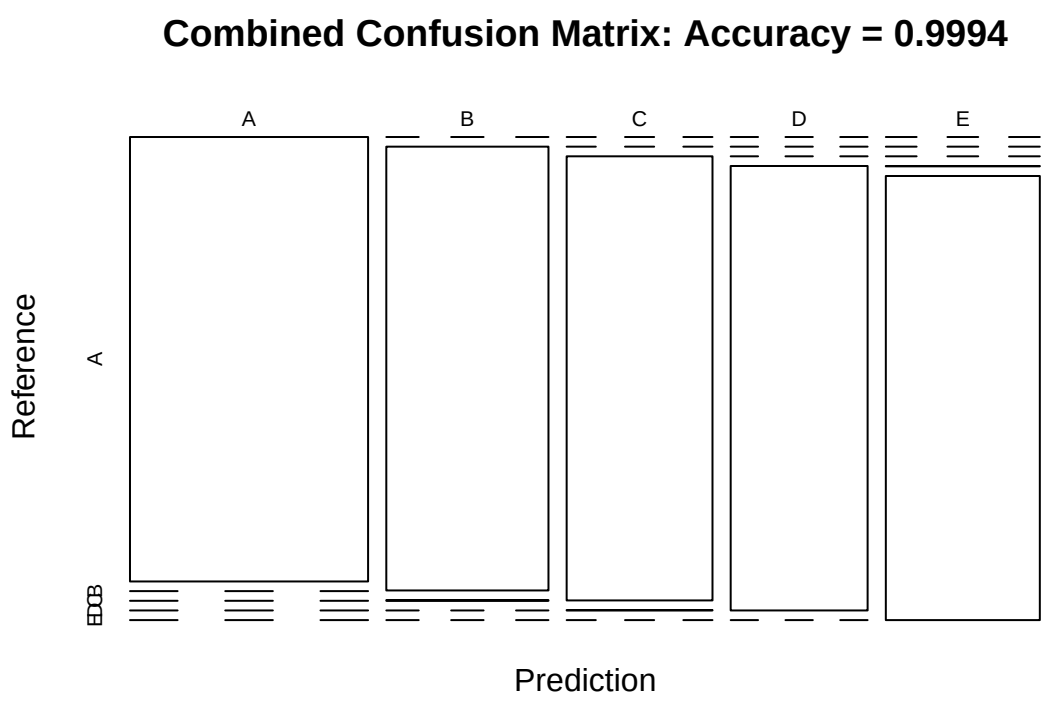
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1953    0    0    0    0
##          B    0 1328    2    0    0
##          C    0    0 1195    1    0
##          D    0    0    0 1123    0
##          E    0    0    0    1 1262
##
## Overall Statistics
##
##                Accuracy : 0.9994
##                  95% CI : (0.9985, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9983   0.9982   1.0000
## Specificity            1.0000   0.9996   0.9998   1.0000   0.9998
## Pos Pred Value         1.0000   0.9985   0.9992   1.0000   0.9992
## Neg Pred Value         1.0000   1.0000   0.9996   0.9997   1.0000
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1934   0.1741   0.1636   0.1838
## Detection Prevalence   0.2845   0.1937   0.1742   0.1636   0.1840
## Balanced Accuracy      1.0000   0.9998   0.9991   0.9991   0.9999
```

```
plot(mod_comb, main = "Combined Model")
```




```
plot(con_comb$table,col = "white", main = paste("Combined Confusion Matrix: Accuracy =", round(con_comb$
```



As we can see, we do have enhanced accuracy as 99.90%. So the combined model is the best model.

5. Apply the Model to Testing Data Set.

5.1. Predict the Testing Data Set Classes

```
tpred_rf<-predict(mod_rf, newdata = testing)
tpred_gbm<-predict(mod_gbm, newdata = testing)
tpred_lda<-predict(mod_lda, newdata = testing)
Mycomb_tpred<-data.frame(pred_rf=tpred_rf, pred_gbm=tpred_gbm, pred_lda=tpred_lda)
tpred_comb<-predict(mod_comb, newdata = Mycomb_tpred)
tpred_comb
```

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

5.2. Write Down File

```
problem_id=1:length(tpred_comb)
prediction_final<-data.frame(problem_id,classe=tpred_comb)
write.table(prediction_final, file = "prediction.csv", sep = ",", row.names = FALSE)
```