Practical Machine Learning project - Evaluate the manner of Weight Lifting exercises

lehhar

18 October 2015

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: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Aim of the project:

The goal of the project is to predict the manner in which the exercise is done. This is the “classe” variable in the training set.

Data:

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

We thank for the allowance to use the data to: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13) . Stuttgart, Germany: ACM SIGCHI, 2013.

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(knitr)
```

Getting and loading the data

The data has been downloaded from the above mentioned source and stored in the working directory.

```
if(!exists("Data")){Data <- read.csv("pml-training.csv")}
if(!exists("DataTest")){DataTest <- read.csv("pml-testing.csv")}
dim(Data); dim(DataTest)
```

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

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

Partitioning the data in a training and testing set:

```
set.seed(9843)
inTrain <- createDataPartition(Data$classe,p=0.6,list=FALSE)
training <- Data[inTrain,]
testing <- Data[-inTrain,]
dim(training); dim(testing); dim(DataTest)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

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

Exploring the data shows that the first 2 columns are only a numbering and names and therefore need to be removed. Another information is that several columns consist of many NA values. Those columns need to be removed too:

```
h <- Data
h <- h[1:20,]
training <- training[,-c(1,2)]
testing <- testing[,-c(1,2)]
DataTest <- DataTest[,-c(1,2)]
h <- h[,-c(1,2)]
for(i in (ncol(training)-1) : 1){
        if((sum(is.na(training[,i])) / nrow(training) > 0.5) || (sum(training[,i] ==
"") > 0.5)){
                training <- training[,-i]
                testing <- testing[,-i]
                DataTest <- DataTest[,-i]
                h <- h[,-i]
        }
}
dim(training); dim(testing); dim(DataTest); dim(h)
```

```
## [1] 11776    58
```

```
## [1] 7846   58
```

```
## [1] 20 58
```

```
## [1] 20 58
```

Remove columns which have variance near 0:

```
nzv <- nearZeroVar(training,saveMetrics=TRUE)
training <- training[,nzv$nzv==FALSE]
testing <- testing[,nzv$nzv==FALSE]
DataTest <- DataTest[,nzv$nzv==FALSE]
h <- h[,nzv$nzv==FALSE]
dim(training); dim(testing); dim(DataTest); dim(h)
```

```
## [1] 11776    57
```

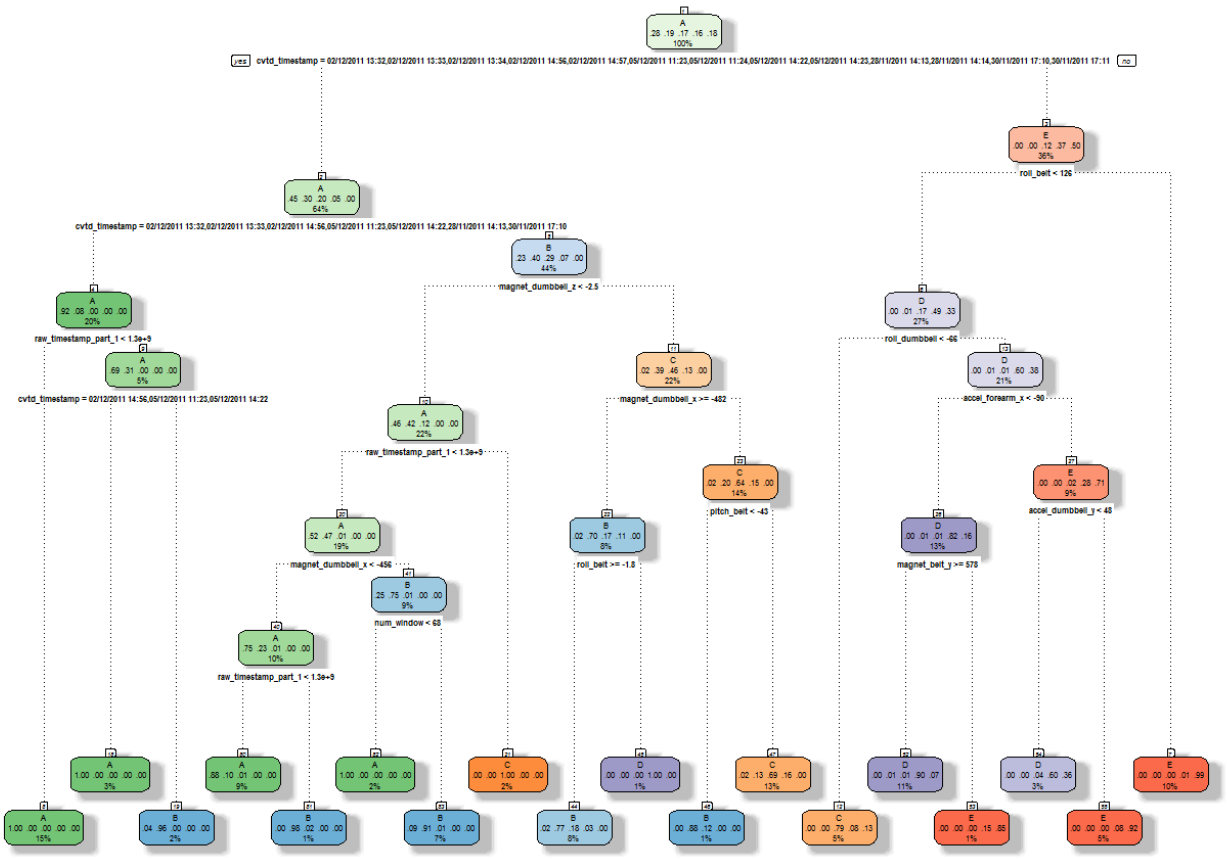
```
## [1] 7846   57
```

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

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

Predicting with Decision Trees

```
decTree <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(decTree)
```



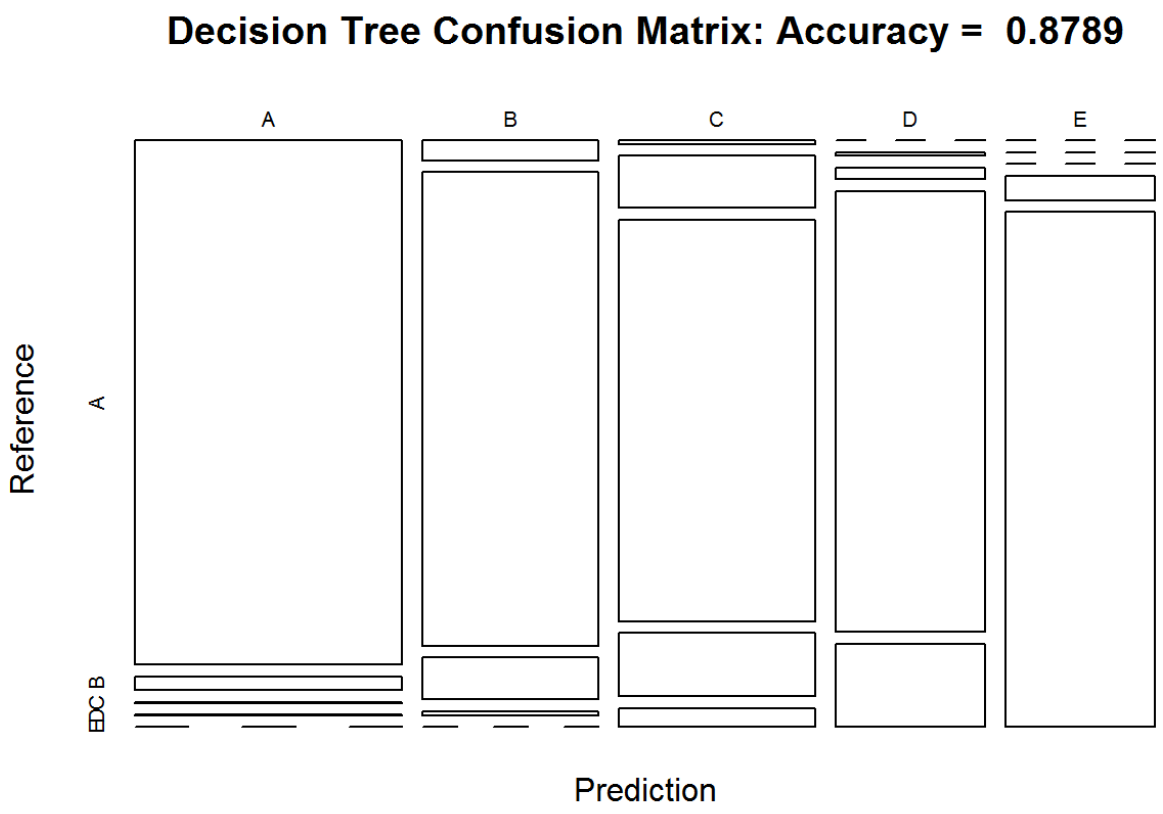
Rattle 2015-Oct-24 19:28:09 Harald

Evaluating the Decision Tree model with the test set:

```
predictionDecTree <- predict(decTree, testing, type="class")
confMatDecTree <- confusionMatrix(predictionDecTree,testing$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2167   57    5    2    0
##          B   54 1294  115   11    0
##          C   11  159 1221  194   57
##          D    0    8   27 1022  193
##          E    0    0    0   57 1192
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8469
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9709   0.8524   0.8925   0.7947   0.8266
## Specificity            0.9886   0.9716   0.9350   0.9652   0.9911
## Pos Pred Value         0.9713   0.8779   0.7436   0.8176   0.9544
## Neg Pred Value         0.9884   0.9648   0.9763   0.9600   0.9621
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2762   0.1649   0.1556   0.1303   0.1519
## Detection Prevalence   0.2843   0.1879   0.2093   0.1593   0.1592
## Balanced Accuracy      0.9797   0.9120   0.9138   0.8800   0.9089
```

```
plot(confMatDecTree$table,col=confMatDecTree$byClass,main=paste("Decision Tree Confusio
n Matrix: Accuracy = ",round(confMatDecTree$overall['Accuracy'],4)))
```

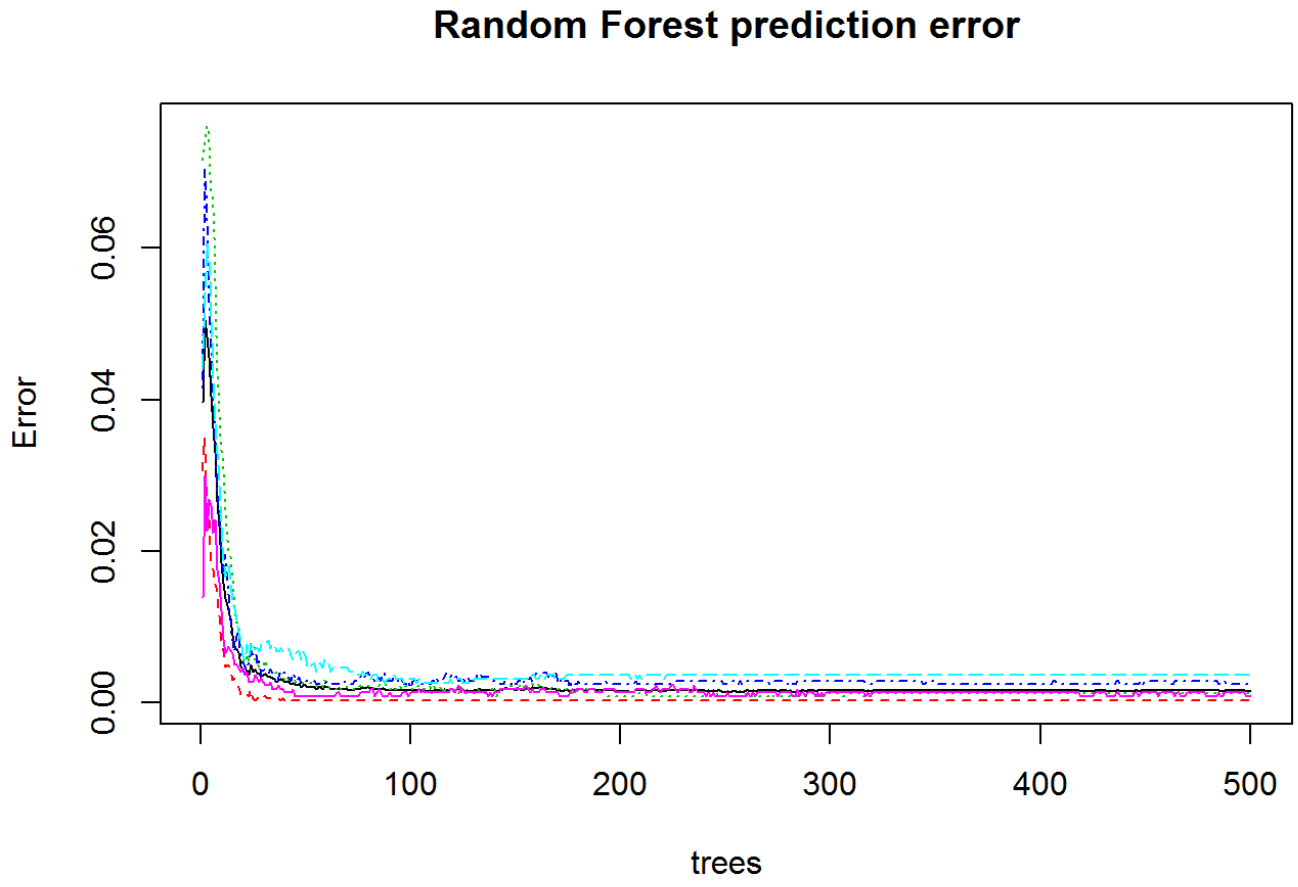


Predicting with Random Forests

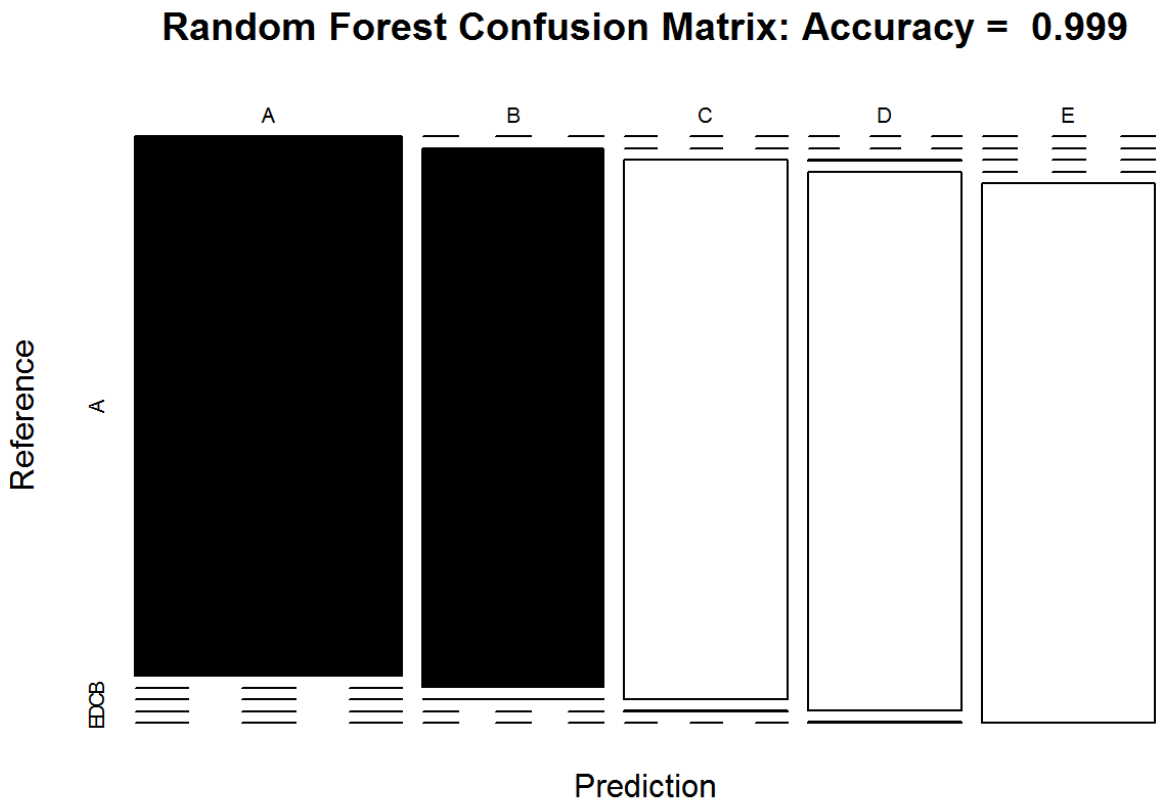
```
randFor <- randomForest(classe ~ .,data=training)
predictionRandFor <- predict(randFor,testing,type="class")
confMatRandFor <- confusionMatrix(predictionRandFor,testing$classe)
confMatRandFor
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    2    0    0
##          C    0    0 1365    2    0
##          D    0    0    1 1284    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9978   0.9984   0.9979
## Specificity            1.0000   0.9997   0.9997   0.9994   1.0000
## Pos Pred Value         1.0000   0.9987   0.9985   0.9969   1.0000
## Neg Pred Value         1.0000   1.0000   0.9995   0.9997   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1740   0.1637   0.1834
## Detection Prevalence   0.2845   0.1937   0.1742   0.1642   0.1834
## Balanced Accuracy      1.0000   0.9998   0.9987   0.9989   0.9990
```

```
plot(randFor,main="Random Forest prediction error")
```




```
plot(confMatRandFor$table,col=confMatRandFor$byClass,main=paste("Random Forest Confusio
n Matrix: Accuracy = ",round(confMatRandFor$overall['Accuracy'],4)))
```



Predicting with Generalized Boosted Regression

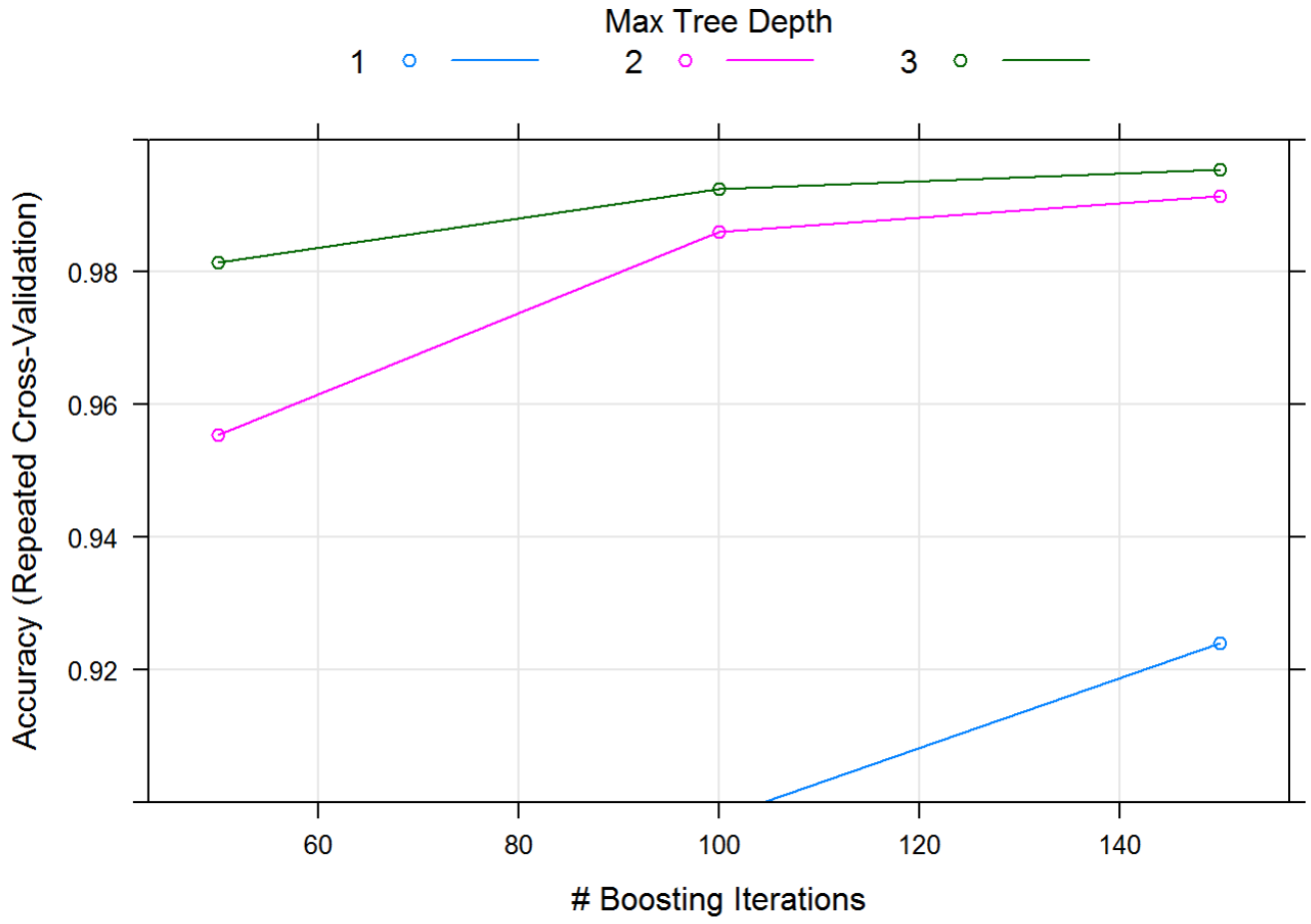
```
fitControl <- trainControl(method="repeatedcv",number=5,repeats=1)
GenBooReg <- train(classe ~ .,data=training,method="gbm",trControl=fitControl,verbose=F
ALSE)
```

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

```
GenBooRegFin <- GenBooReg$finalModel
predictionGenBooRegFin <- predict(GenBooReg,newdata=testing)
GenBooRegAccuracyTest <- confusionMatrix(predictionGenBooRegFin,testing$classe)
GenBooRegAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1514    2    0    0
##          C    0    2 1357    3    0
##          D    0    2    9 1283    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.9962, 0.9985)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9969
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9920   0.9977   0.9993
## Specificity            1.0000   0.9997   0.9992   0.9982   1.0000
## Pos Pred Value         1.0000   0.9987   0.9963   0.9907   1.0000
## Neg Pred Value         1.0000   0.9994   0.9983   0.9995   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1730   0.1635   0.1837
## Detection Prevalence   0.2845   0.1932   0.1736   0.1651   0.1837
## Balanced Accuracy      1.0000   0.9985   0.9956   0.9979   0.9997
```

```
plot(GenBooReg,ylim=c(0.9,1))
```



Applying the best fitted model to the test data set

Random Forests gave 99.9% accuracy for the testing data set which is higher than the accuracy for Decision Tree and the Generalized Boosted Regression.

The expected out-of-sample error is 100% - 99.9% = 0.1%

```
for(i in 1:20){
        for(j in 1:ncol(h)){
                h[i,j] <- DataTest[i,j]
        }
}
TestPrediction <- predict(randFor,h,type="class")
TestPrediction
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
# Write the results to a text file for submission
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}

# pml_write_files(TestPrediction)
```