Practical Machine Learning Project

Jeho Kang

February 4, 2018

Project Introduction

Backgroud

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). ### 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) The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (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. ### Project 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.

Loading and cleaning data

```
pml_train = read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv"))
pml_test = read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testi
ng.csv"))
```

Remove the columns having at least 90% of NA or blank values on the training and test datasets

```
colToRemove = which(colSums(is.na(pml_train) | pml_train=="") > 0.9*dim(pml_train)[1])
pml_train = pml_train[,-colToRemove]

colToRemove = which(colSums(is.na(pml_test) | pml_test=="") > 0.9*dim(pml_test)[1])
pml_test = pml_test[,-colToRemove]
```

Remove the first five variables that don't make intuitive sense for prediction (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp)

```
pml_train = pml_train[,-(1:5)]
pml_test = pml_test[,-(1:5)]
```

Model Building

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
library(rattle)
```

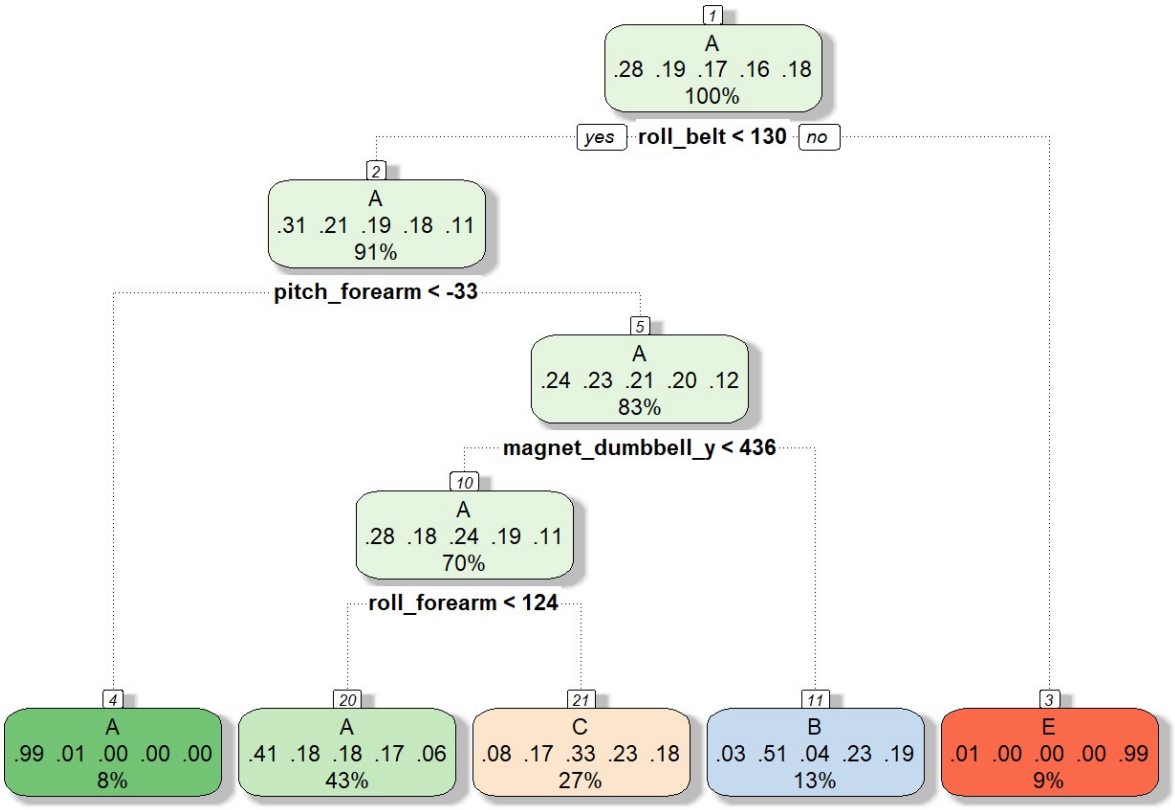
```
## Warning: package 'rattle' was built under R version 3.4.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(1234)
inTrain = createDataPartition(y=pml_train$classe, p=0.7,list=F)
trainset = pml_train[inTrain,]
testset = pml_train[-inTrain,]
```

1) Decision Tree Model

```
set.seed(1234)
modFitDecTree = train(classe ~ ., data=trainset, method="rpart")
fancyRpartPlot(modFitDecTree$finalModel)
```

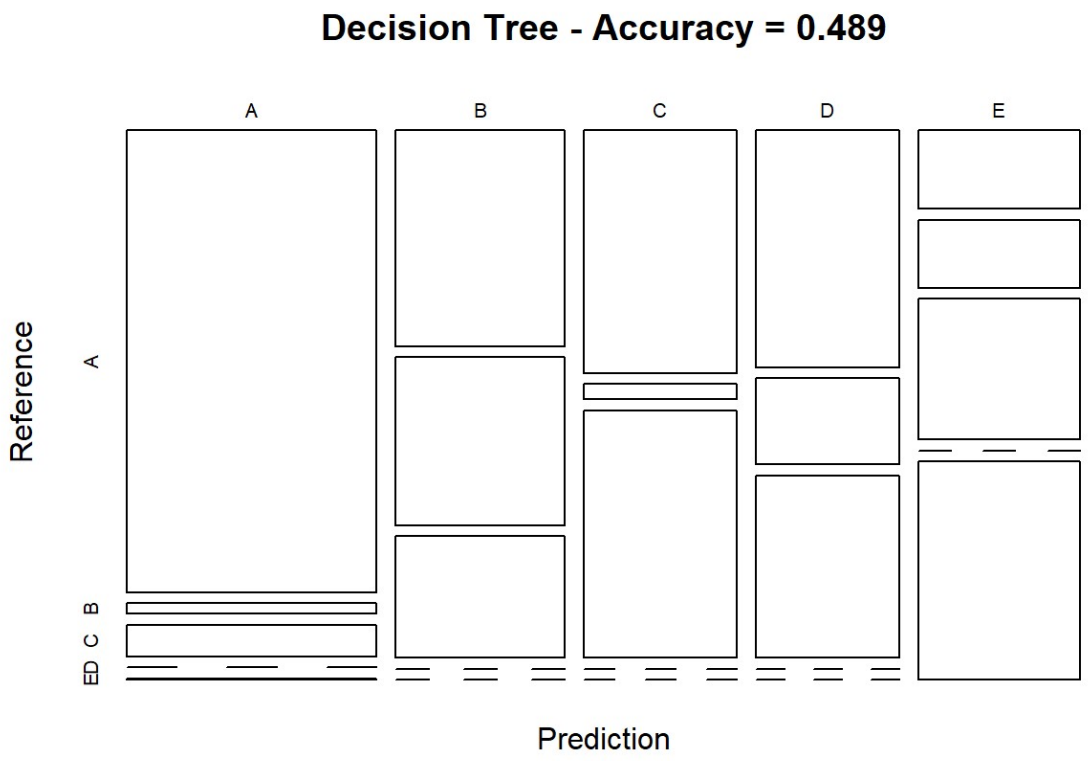


Rattle 2018-Feb-04 16:28:39 jeho_

```
predDecTree = predict(modFitDecTree, newdata=testset)
confMDecTree = confusionMatrix(testset$classe,predDecTree)
confMDecTree
```

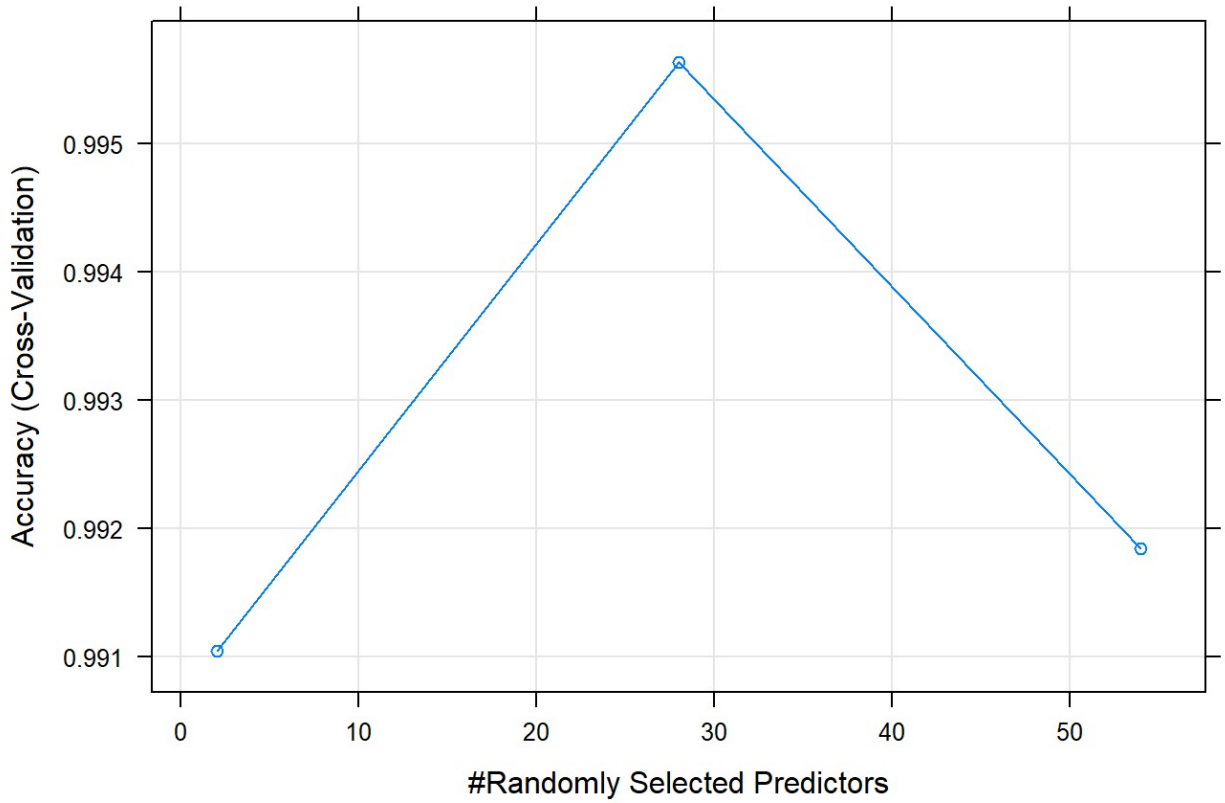
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530   35  105    0    4
##          B  486  379  274    0    0
##          C  493   31  502    0    0
##          D  452  164  348    0    0
##          E  168  145  302    0  467
##
## Overall Statistics
##
##                Accuracy : 0.489
##                  95% CI : (0.4762, 0.5019)
##     No Information Rate : 0.5317
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3311
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4890   0.5027   0.3279       NA  0.99151
## Specificity            0.9478   0.8519   0.8797   0.8362  0.88641
## Pos Pred Value         0.9140   0.3327   0.4893       NA  0.43161
## Neg Pred Value         0.6203   0.9210   0.7882       NA  0.99917
## Prevalence             0.5317   0.1281   0.2602   0.0000  0.08003
## Detection Rate         0.2600   0.0644   0.0853   0.0000  0.07935
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638  0.18386
## Balanced Accuracy      0.7184   0.6773   0.6038       NA  0.93896
```

```
plot(confMDecTree$table, col = confMDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMDecTree$overall['Accuracy'],4)))
```



2) Random Forests Model

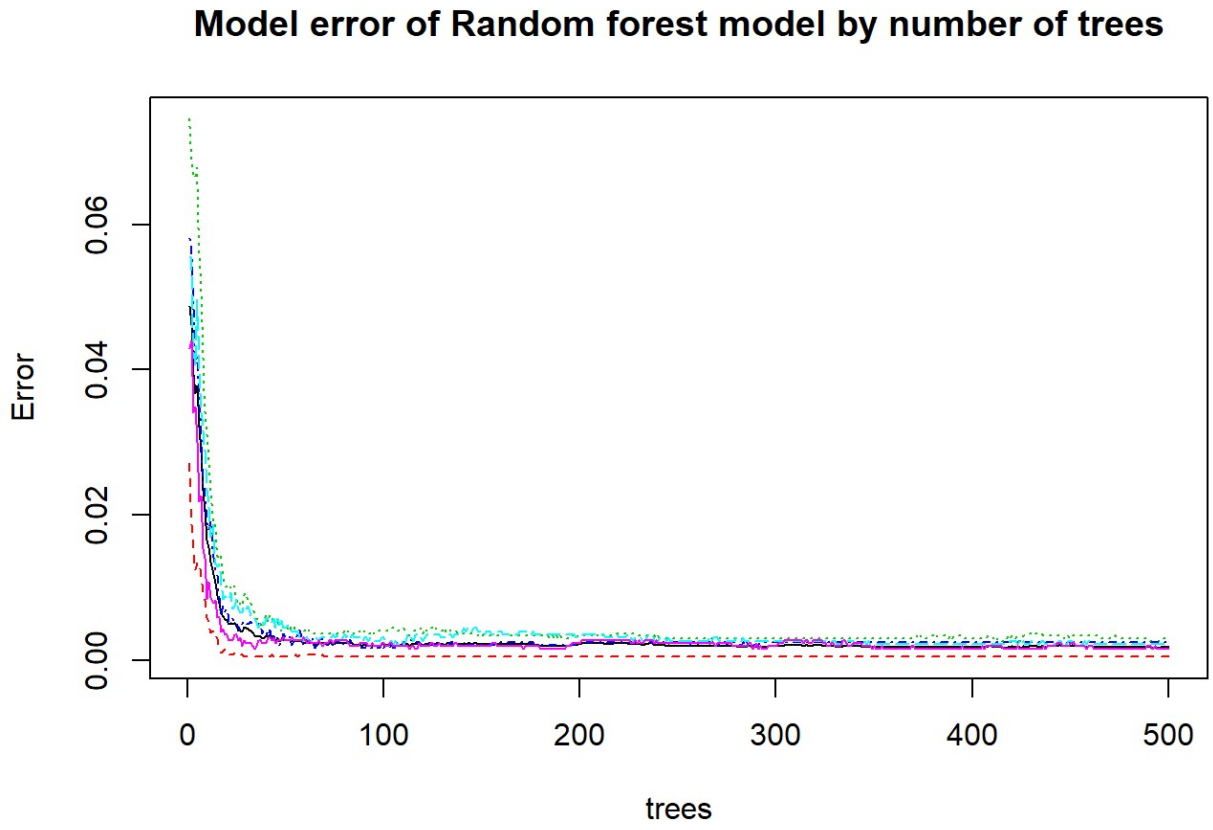
```
set.seed(1234)
modFitRF = train(classe ~ ., data=trainset, method="rf",
                 trControl = trainControl(method="cv", number=3),
                 verbose=F)
plot(modFitRF, main="Accuracy of Random forest model by number of predictors")
```




```
names(modFitRF$finalModel)
```

```
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
## [19] "xNames"          "problemType"     "tuneValue"
## [22] "obsLevels"       "param"
```

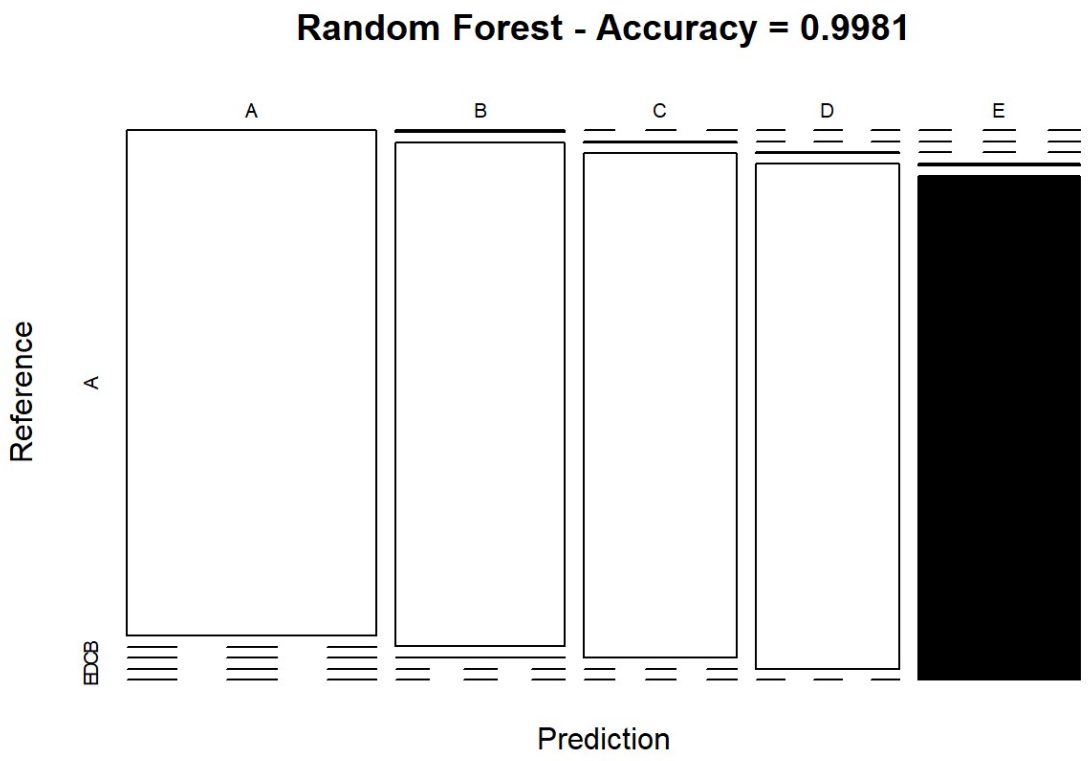
```
plot(modFitRF$finalModel,main="Model error of Random forest model by number of trees")
```




```
predRF = predict(modFitRF, newdata=testset)
confMRF = confusionMatrix(testset$classe,predRF)
confMRF
```

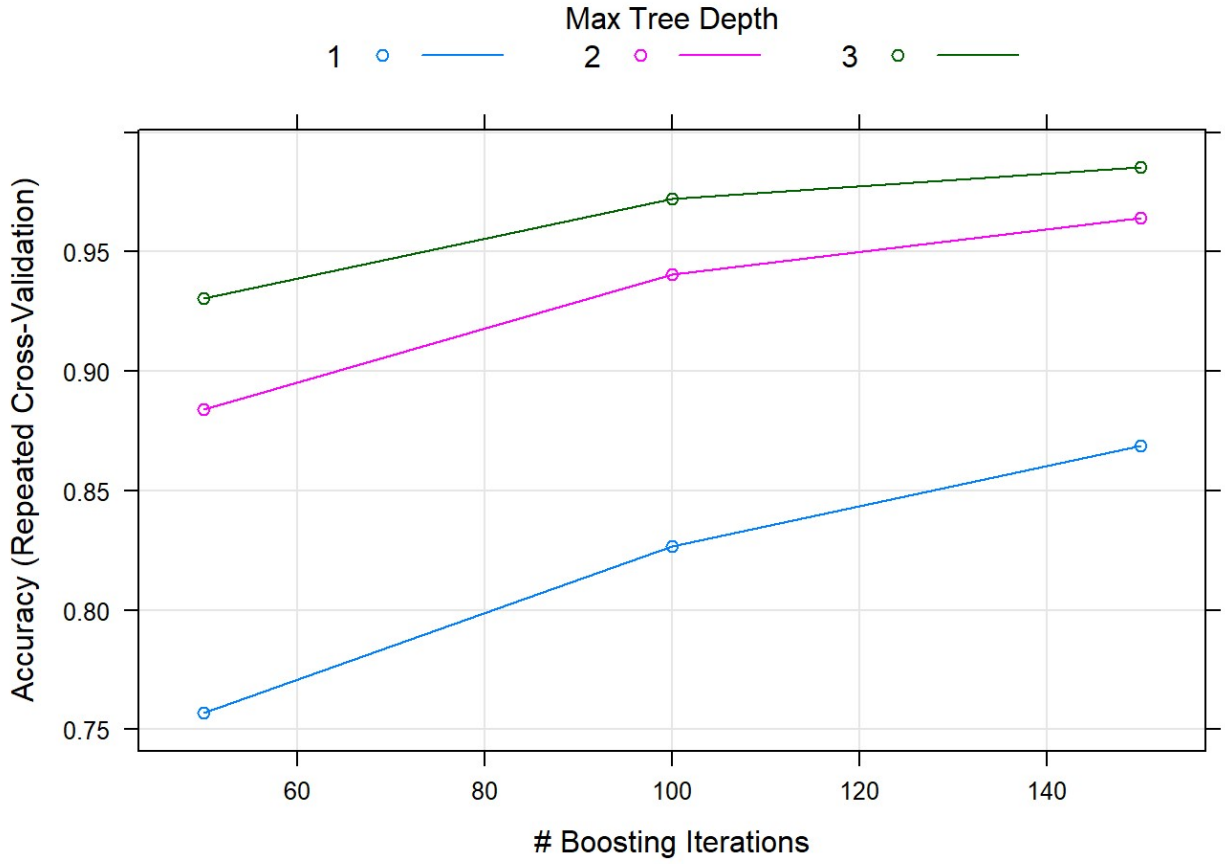
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    3 1135    1    0    0
##          C    0    2 1024    0    0
##          D    0    0    1  963    0
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9982   0.9981   0.9959   1.0000
## Specificity            1.0000   0.9992   0.9996   0.9998   0.9992
## Pos Pred Value         1.0000   0.9965   0.9981   0.9990   0.9963
## Neg Pred Value         0.9993   0.9996   0.9996   0.9992   1.0000
## Prevalence             0.2850   0.1932   0.1743   0.1643   0.1832
## Detection Rate         0.2845   0.1929   0.1740   0.1636   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9991   0.9987   0.9988   0.9978   0.9996
```

```
plot(confMRF$table, col = confMRF$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMRF$overall['Accuracy'],4)))
```



3) Generalized Boosting Model (GBM)

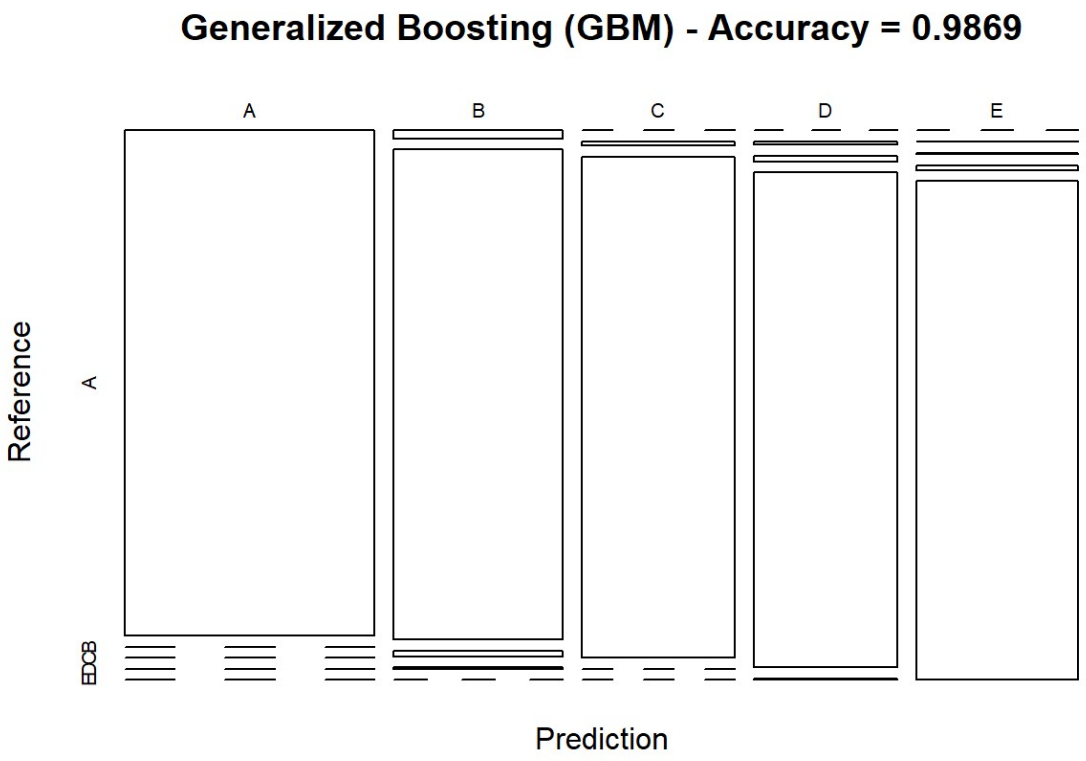
```
set.seed(1234)
modFitGBM = train(classe ~ ., data=trainset, method="gbm",
                  trControl = trainControl(method = "repeatedcv",
                                           number = 5, repeats = 1),
                  verbose=F)
plot(modFitGBM)
```




```
predGBM = predict(modFitGBM, newdata=testset)
confMGBM = confusionMatrix(testset$classe,predGBM)
confMGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B   18 1105   13    3    0
##          C    0    9 1017    0    0
##          D    0    6   11  944    3
##          E    0    1    3   10 1068
##
## Overall Statistics
##
##                Accuracy : 0.9869
##                  95% CI : (0.9837, 0.9897)
##     No Information Rate : 0.2875
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9834
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9894   0.9857   0.9741   0.9864   0.9972
## Specificity            1.0000   0.9929   0.9981   0.9959   0.9971
## Pos Pred Value         1.0000   0.9701   0.9912   0.9793   0.9871
## Neg Pred Value         0.9957   0.9966   0.9944   0.9974   0.9994
## Prevalence             0.2875   0.1905   0.1774   0.1626   0.1820
## Detection Rate         0.2845   0.1878   0.1728   0.1604   0.1815
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9947   0.9893   0.9861   0.9912   0.9971
```

```
plot(confMGBM$table, col = confMGBM$byClass,
     main = paste("Generalized Boosting (GBM) - Accuracy =",
                  round(confMGBM$overall['Accuracy'],4)))
```



Conclusion

The Random Forests model is showing best accuracy and this model will be used for predicting the classe values for the test set.

```
final_pred = predict(modFitRF, newdata=pml_test)
final_pred
```

```
##  [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
```