Prediction Assignment Writeup

Merana

2024-09-03

Created with knitr

Overview

This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science. It was built up in RStudio, using its knitr functions, meant to be published in html format. This analysis meant to be the basis for the course quiz and a prediction assignment writeup. The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

Data Loading and Exploratory Analysis

Data Collection (Loading) 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.

Libraries

```
set.seed(322)
library(knitr)
library(lattice)
library(ggplot2)
# install.packages("caret", dependencies = TRUE)
library(caret)
# install.packages("rpart", dependencies = TRUE)
library(rpart)
# install.packages("rpart.plot", dependencies = TRUE)
library(rpart.plot)
# install.packages("corrplot", dependencies = TRUE)
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```
library(RColorBrewer)
# install.packages("rattle", dependencies = TRUE)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(data.table)
library(corrplot)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
# install.packages("gbm", dependencies = TRUE)
library(gbm)
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-dev
elopers/gbm3
```

Data loading

```
path <- getwd()
# download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#               , destfile = paste(path, "pml-training.csv", sep = "/"))
# download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
              # , destfile = paste(path, "pml-testing.csv", sep = "/"))
data_train <- read.csv("pml-training.csv")
data_quiz <- read.csv("pml-testing.csv")
dim(data_train)
```

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

```
dim(data_quiz)
```

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

Some columns were identified as character, although they are numeric. Let's fix those columns.

```
data_train[, 12:159] <- lapply(data_train[, 12:159], as.numeric)
```

```
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
## Warning in lapply(data_train[, 12:159], as.numeric): NAs introduced by coercion
```

```
data_quiz[, 12:159] <- lapply(data_quiz[, 12:159], as.numeric)
```

Splitting data into train and test.

```
in_train  <- createDataPartition(data_train$classe, p=0.70, list=FALSE)
train_set <- data_train[ in_train, ]
test_set  <- data_train[-in_train, ]
```

The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```
nzv_var <- nearZeroVar(train_set)
train_set <- train_set[ , -nzv_var]
test_set  <- test_set [ , -nzv_var]
dim(train_set)
```

```
## [1] 13737   128
```

```
dim(test_set)
```

```
## [1] 5885  128
```

Remove variables that are mostly NA. A threshlod of 95 % is selected

```
na_var <- sapply(train_set, function(x) mean(is.na(x))) > 0.95
train_set <- train_set[ , na_var == FALSE]
test_set  <- test_set [ , na_var == FALSE]
dim(train_set)
```

```
## [1] 13737    59
```

```
dim(test_set)
```

```
## [1] 5885   59
```

Since columns 1 to 5 are identification variables only, they will be removed as well

```
train_set <- train_set[ , -(1:5)]
test_set  <- test_set [ , -(1:5)]
dim(train_set)
```

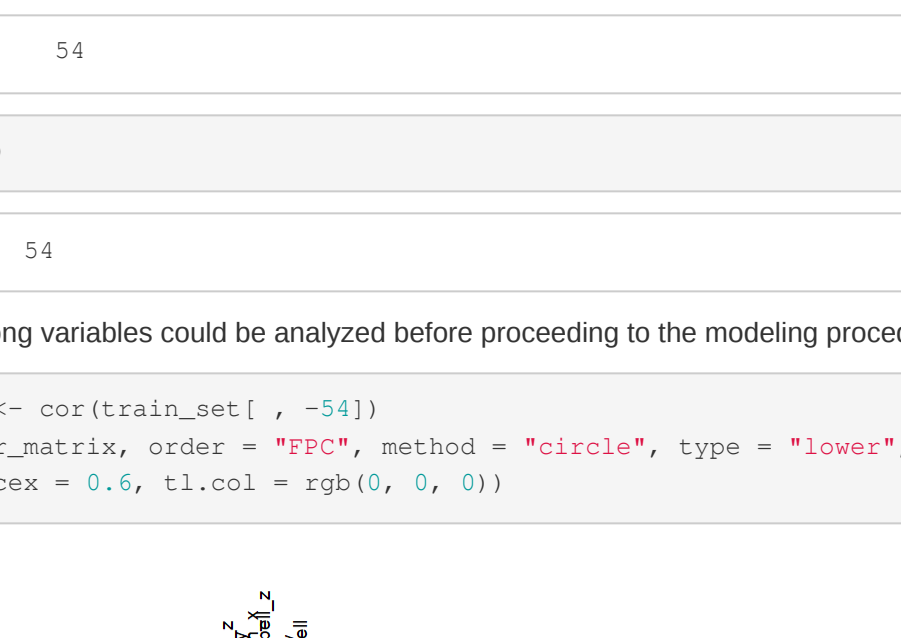
```
## [1] 13737    54
```

```
dim(test_set)
```

```
## [1] 5885   54
```

A correlation among variables could be analyzed before proceeding to the modeling procedures.

```
corr_matrix <- cor(train_set[ , -54])
corrplot(corr_matrix, order = "FPC", method = "circle", type = "lower",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



Choosing a model

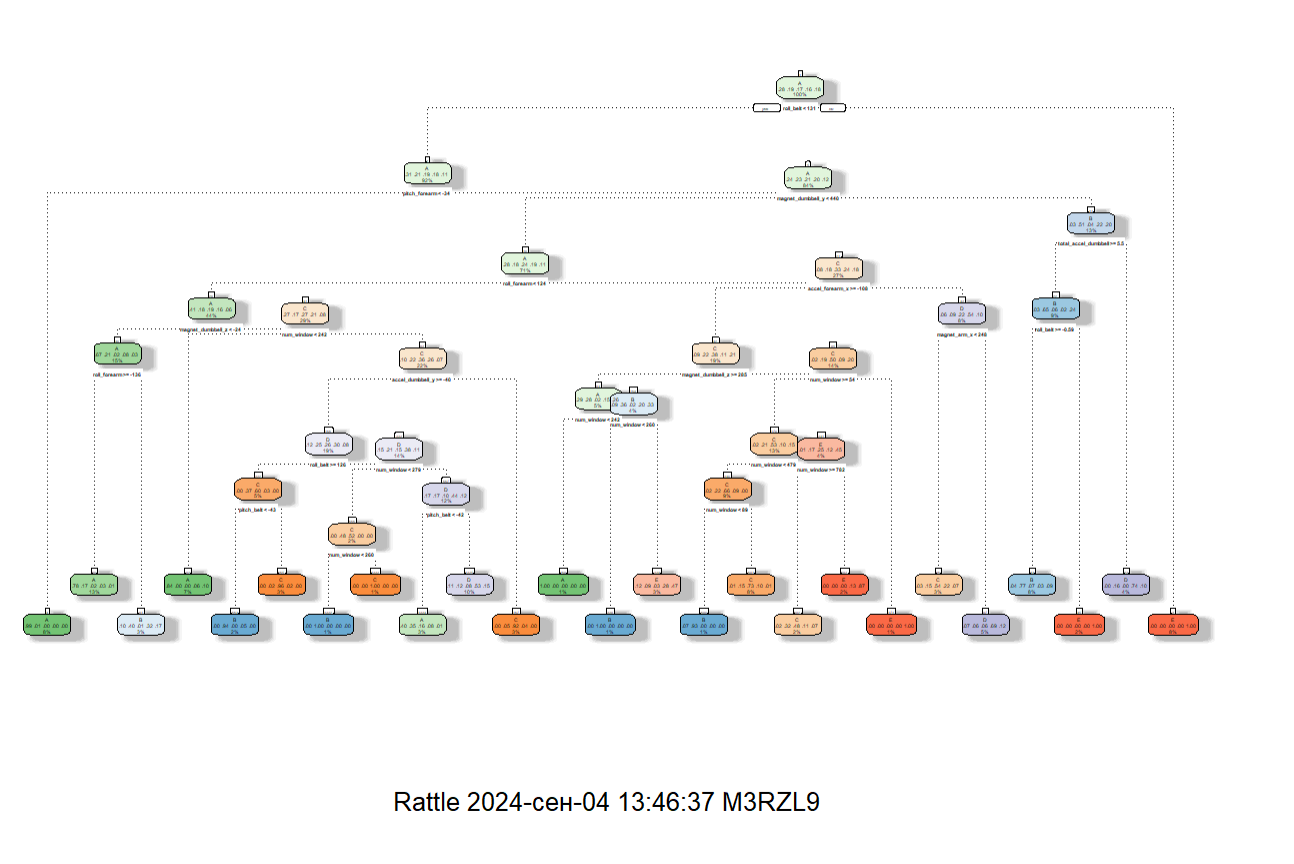
Three methods will be applied to model the regressions (in the Train data set) and the best one (with higher accuracy when applied to the Test data set) will be used for the quiz predictions. The methods are: - Decision Tree, - Generalized Boosted Model (GBM) and - Random Forests (rf), as described below.

Training the Model A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

Decision Tree Model

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

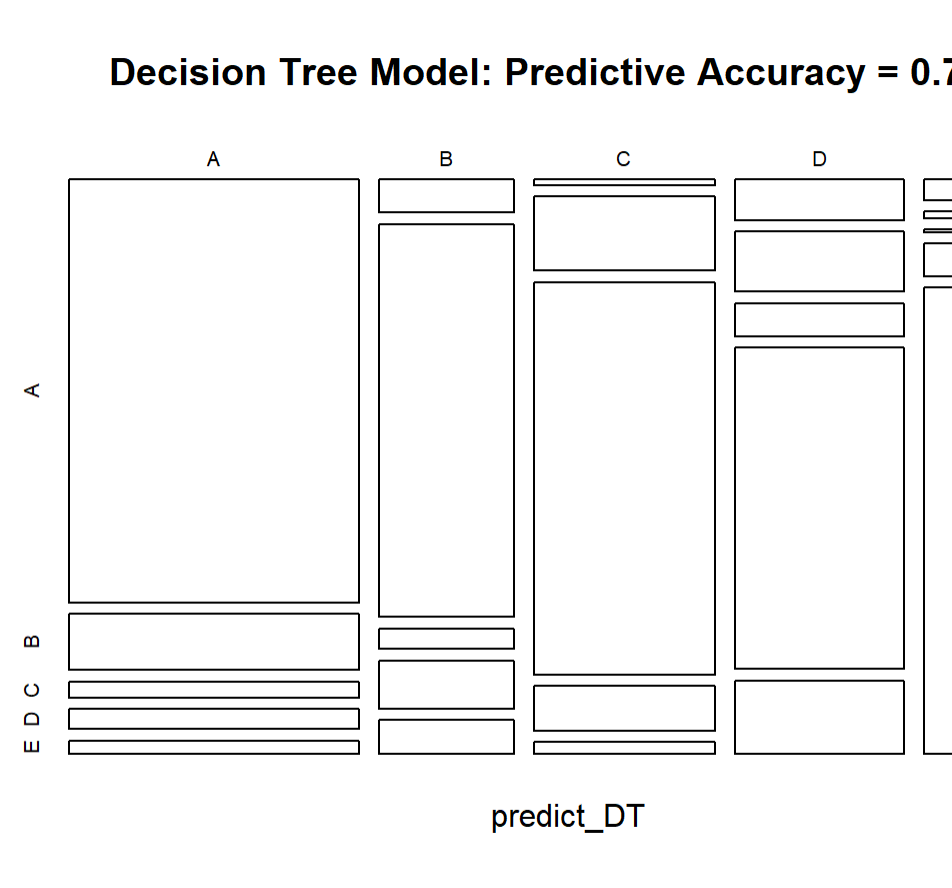
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```




```
predict_DT <- predict(fit_DT, newdata = test_set, type="class")
conf_matrix_DT <- confusionMatrix(table(predict_DT, test_set$classe))
conf_matrix_DT
```

```
## Confusion Matrix and Statistics
##
##
## predict_DT    A    B    C    D    E
##          A 1489  198   57   70   48
##          B   54  643   34   78   56
##          C   12  163  862   98   26
##          D   84  124   68  663  152
##          E   35   11    5   55  800
##
## Overall Statistics
##
##                Accuracy : 0.7573
##                  95% CI : (0.7462, 0.7683)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6923
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8895   0.5645   0.8402   0.6878   0.7394
## Specificity            0.9114   0.9532   0.9385   0.9130   0.9779
## Pos Pred Value         0.7997   0.7434   0.7425   0.6077   0.8830
## Neg Pred Value         0.9540   0.9012   0.9653   0.9372   0.9434
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2530   0.1093   0.1465   0.1127   0.1359
## Detection Prevalence   0.3164   0.1470   0.1973   0.1854   0.1540
## Balanced Accuracy      0.9005   0.7589   0.8893   0.8004   0.8587
```

```
plot(conf_matrix_DT$table, col = conf_matrix_DT$byClass,
     main = paste("Decision Tree Model: Predictive Accuracy =",
                  round(conf_matrix_DT$overall['Accuracy'], 4)))
```



Generalized Boosted Model (GBM)

```
ctrl_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_GBM  <- train(classe ~ ., data = train_set, method = "gbm",
                  trControl = ctrl_GBM, verbose = FALSE)
fit_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predict_GBM <- predict(fit_GBM, newdata = test_set)
conf_matrix_GBM <- confusionMatrix(table(predict_GBM, test_set$classe))
conf_matrix_GBM
```

```
## Confusion Matrix and Statistics
##
##
## predict_GBM    A    B    C    D    E
##           A 1674   19    0    1    0
##           B    0 1105    5    4    7
##           C    0   14 1016   12    1
##           D    0    1    4  947   17
##           E    0    0    1    0 1057
##
## Overall Statistics
##
##                Accuracy : 0.9854
##                  95% CI : (0.982, 0.9883)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9815
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9701   0.9903   0.9824   0.9769
## Specificity            0.9953   0.9966   0.9944   0.9955   0.9998
## Pos Pred Value         0.9882   0.9857   0.9741   0.9773   0.9991
## Neg Pred Value         1.0000   0.9929   0.9979   0.9965   0.9948
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1878   0.1726   0.1609   0.1796
## Detection Prevalence   0.2879   0.1905   0.1772   0.1647   0.1798
## Balanced Accuracy      0.9976   0.9834   0.9923   0.9889   0.9883
```

Random Forest

```
ctrl_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RF  <- train(classe ~ ., data = train_set, method = "rf",
                 trControl = ctrl_RF, verbose = FALSE)
fit_RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2649    2    0    0 0.0033860045
## C    0    6 2388    2    0 0.0033388982
## D    0    0    7 2245    0 0.0031083481
## E    0    1    0    6 2518 0.0027722772
```

```
predict_RF <- predict(fit_RF, newdata = test_set)
conf_matrix_RF <- confusionMatrix(table(predict_RF, test_set$classe))
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##
## predict_RF    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1135    1    0    0
##          C    0    0 1025    2    0
##          D    0    0    0  961    2
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2845
##     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            0.9994   0.9965   0.9990   0.9969   0.9982
## Specificity            0.9991   0.9996   0.9996   0.9996   0.9998
## Pos Pred Value         0.9976   0.9982   0.9981   0.9979   0.9991
## Neg Pred Value         0.9998   0.9992   0.9998   0.9994   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1742   0.1633   0.1835
## Detection Prevalence   0.2850   0.1932   0.1745   0.1636   0.1837
## Balanced Accuracy      0.9992   0.9980   0.9993   0.9982   0.9990
```

Evaluate the Model (Fitting models)

Applying the Best Predictive Model to the Test Data To summarize, the predictive accuracy of the three models evaluated is as follows:

Summary of the results: - Decision tree model - is the worst model running, has the low mean and the highest standard deviation. - GBM model - has a decent mean accuracy but a little bit lower accuracy than RF. - Random Fores model - has the highest mean accuracy and lowest standard deviation

Parameter Tuning Checking prediction accuracy on my own testing/validation set. I am expecting similar accuracy as the mean from the cross validation.

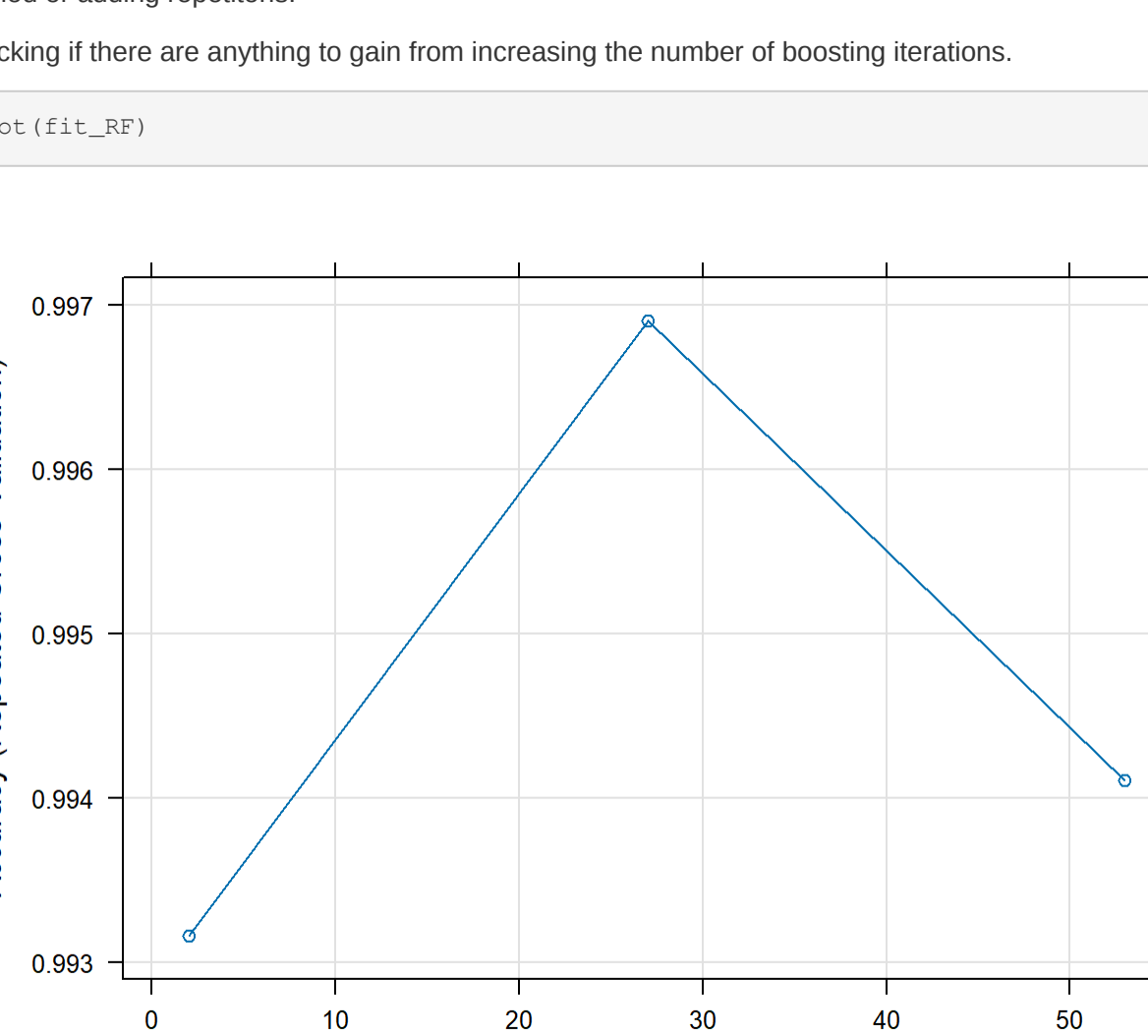
The kappa statistic (labeled Kappa in the previous output) adjusts accuracy by accounting for the possibility of a correct prediction by chance alone. Kappa values range to a maximum value of 1, which indicates perfect agreement between the model's predictions and the true values—a rare occurrence. Values less than one indicate imperfect agreement.

Depending on how your model is to be used, the interpretation of the kappa statistic might vary. One common interpretation is shown as follows: • Poor agreement = Less than 0.20 • Fair agreement = 0.20 to 0.40 • Moderate agreement = 0.40 to 0.60 • Good agreement = 0.60 to 0.80 • Very good agreement = 0.80 to 1.00

This three models preforms as expected, the deviation from the cross validation accuracy is low and I do not see a reason to change resampling method or adding repetitons.

Checking if there are anything to gain from increasing the number of boosting iterations.

```
plot(fit_RF)
```




```
print(fit_RF$bestTune)
```

```
##   mtry
## 2   27
```

The predictive accuracy of the Random Forest model is excellent at 99.8 %. Accuracy has plateaued, and further tuning would only yield decimal gain. - The best tuning parameters hads 150 trees (boosting iterations), - interaction depth 3 - shrinkage 0.1.

Make Predictions Deciding to predict with this model.

Decision Tree Model: 77.5% Generalized Boosted Model: 98.67% Random Forest Model: 99.72%

The Random Forest model is selected and applied to make predictions on the 20 data points from the original testing dataset (data_quiz).

```
cat("Predictions: ", paste(predict(fit_RF, data_quiz)))
```

```
## Predictions:  B A B A A E D B A A B C B A E E A B B B
```