MainPML.R

rstudio

2022-08-18

```
# final project assignment Anca Maria Nagy
#Loading all the libraries and the data

library(lattice)
library(ggplot2)
library(caret)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

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

```
library(rattle)
```

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

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

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## 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(corrplot)
```

```
## corrplot 0.90 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(12345)

train_in <- read.csv("./data/pml-training.csv")
valid_in <- read.csv("./data/pml-testing.csv")

dim(train_in)
```

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

```
dim(valid_in)
```

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

```
#split the training set into a validation and sub training set.

trainData<- train_in[, colSums(is.na(train_in)) == 0]
validData <- valid_in[, colSums(is.na(valid_in)) == 0]
dim(trainData)
```

```
## [1] 19622    93
```

```
trainData<- train_in[, colSums(is.na(train_in)) == 0]
validData <- valid_in[, colSums(is.na(valid_in)) == 0]
dim(trainData)
```

```
## [1] 19622    93
```

```
trainData <- trainData[, -c(1:7)]
validData <- validData[, -c(1:7)]
dim(trainData)
```

```
## [1] 19622    86
```

```
dim(validData)
```

```
## [1] 20 53
```

```
set.seed(12345)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[inTrain, ]
testData <- trainData[-inTrain, ]
dim(trainData)
```

```
## [1] 13737    86
```

```
dim(testData)
```

```
## [1] 4140   86
```

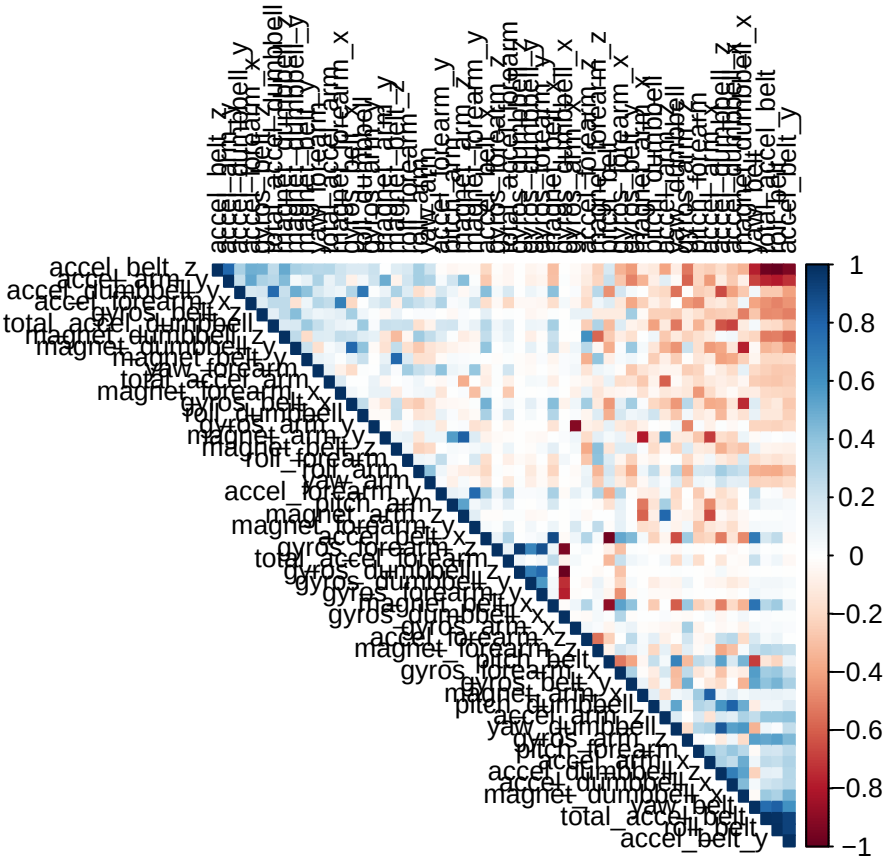
```
# Create and Test the Models
NonZeroV <- nearZeroVar(trainData)
trainData <- trainData[, -NonZeroV]
testData  <- testData[, -NonZeroV]
dim(trainData)
```

```
## [1] 13737    53
```

```r
dim(testData)
```

```
## [1] 4140   53
```

```r
cor_mat <- cor(trainData[, -53])
corrplot(cor_mat, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



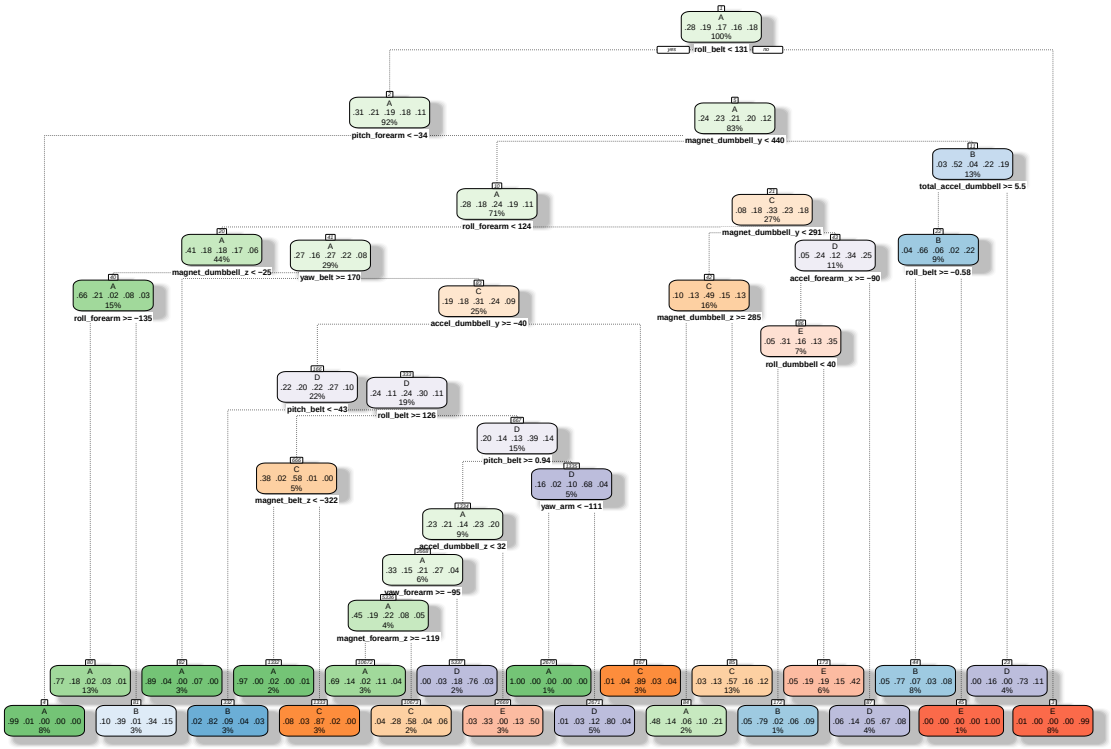

```r
highlyCorrelated = findCorrelation(cor_mat, cutoff=0.75)

names(trainData)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "accel_arm_y"       "total_accel_belt"  "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"        "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "accel_dumbbell_x"
## [13] "accel_arm_x"       "accel_arm_z"       "magnet_arm_y"
## [16] "magnet_belt_z"     "accel_forearm_y"   "gyros_forearm_y"
## [19] "gyros_dumbbell_x"  "gyros_dumbbell_z"  "gyros_arm_x"
```

```r
#Model building

# 1. Classification tree
set.seed(12345)
decisionTreeMod1 <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(decisionTreeMod1)
```



Rattle 2022−Aug−18 10:12:41 rstudio

```
predictTreeMod1 <- predict(decisionTreeMod1, testData, type = "class")

cmtree <- confusionMatrix(predictTreeMod1, as.factor(testData$classe))
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1086  134   12   46   28
##          B   30  438   40   42   54
##          C   27  103  589  106   81
##          D   11   53   43  406   40
##          E   18   70   39   53  591
##
## Overall Statistics
##
##                Accuracy : 0.7512
##                  95% CI : (0.7377, 0.7643)
##     No Information Rate : 0.2831
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6841
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9266   0.5489   0.8147  0.62175   0.7443
## Specificity            0.9259   0.9503   0.9072  0.95784   0.9462
## Pos Pred Value         0.8315   0.7252   0.6501  0.73418   0.7665
## Neg Pred Value         0.9697   0.8982   0.9586  0.93114   0.9397
## Prevalence             0.2831   0.1928   0.1746  0.15773   0.1918
## Detection Rate         0.2623   0.1058   0.1423  0.09807   0.1428
## Detection Prevalence   0.3155   0.1459   0.2188  0.13357   0.1862
## Balanced Accuracy      0.9262   0.7496   0.8609  0.78979   0.8453
```

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

# 2. Prediction with Random Forest

controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=controlRF)
modRF1$finalModel
```

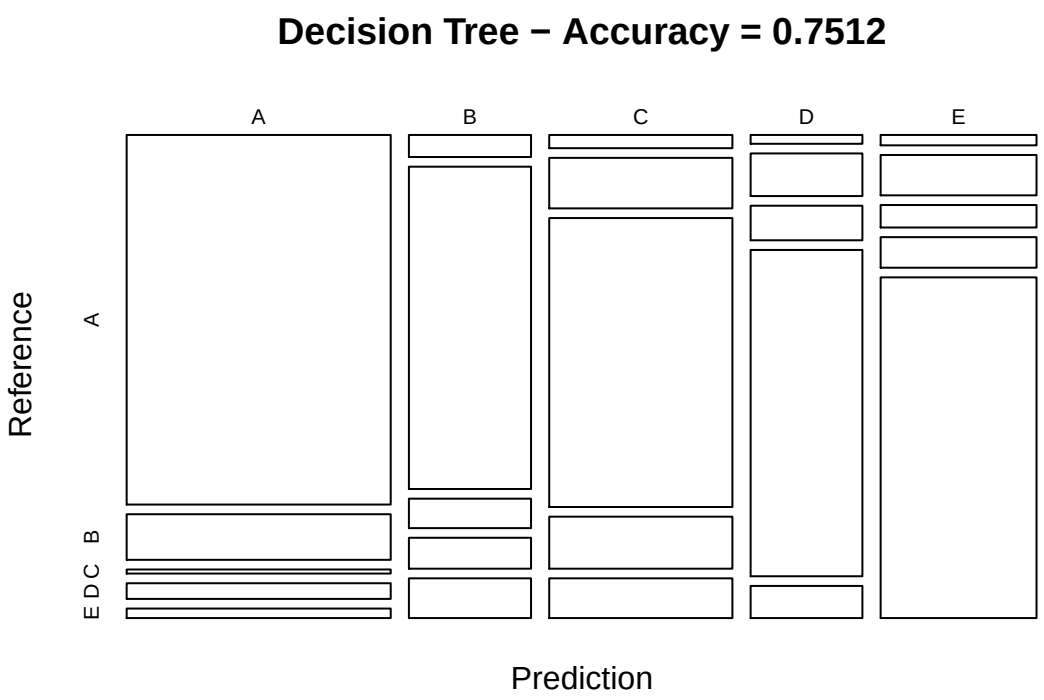
```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    4    1    0    2 0.001792115
## B   21 2630    7    0    0 0.010534236
## C    0   17 2372    7    0 0.010016694
## D    1    1   18 2230    2 0.009769094
## E    0    2    5    5 2513 0.004752475
```

```
predictRF1 <- predict(modRF1, newdata=testData)
cmrf <- confusionMatrix(predictRF1, as.factor(testData$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1172    0    0    0    0
##          B    0  798    0    0    0
##          C    0    0  723    0    0
##          D    0    0    0  653    0
##          E    0    0    0    0  794
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2831
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Rate         0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Prevalence   0.2831   0.1928   0.1746   0.1577   0.1918
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

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




```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=controlRF)
modRF1$finalModel
```

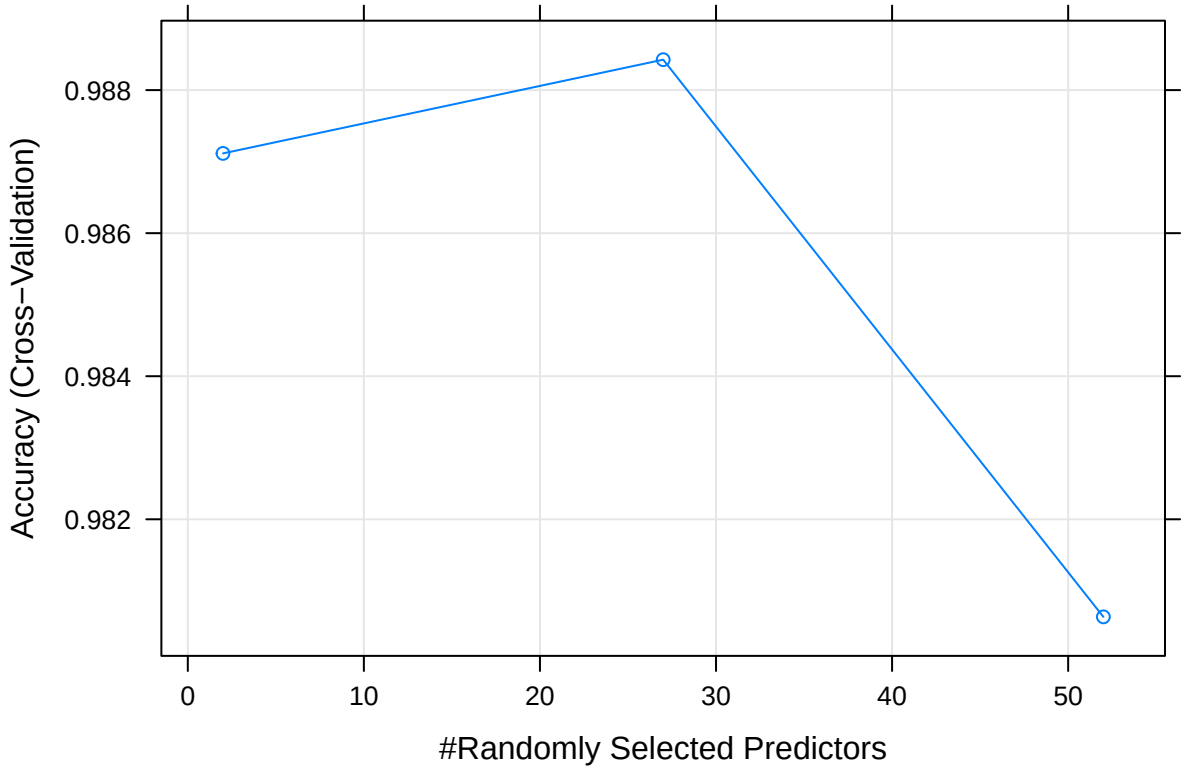
```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
```

```
##      A    B    C    D    E class.error
## A 3899    5    0    0    2 0.001792115
## B   18 2633    7    0    0 0.009405568
## C    0   15 2373    8    0 0.009599332
## D    0    2   22 2225    3 0.011989343
## E    0    2    5    7 2511 0.005544554
```

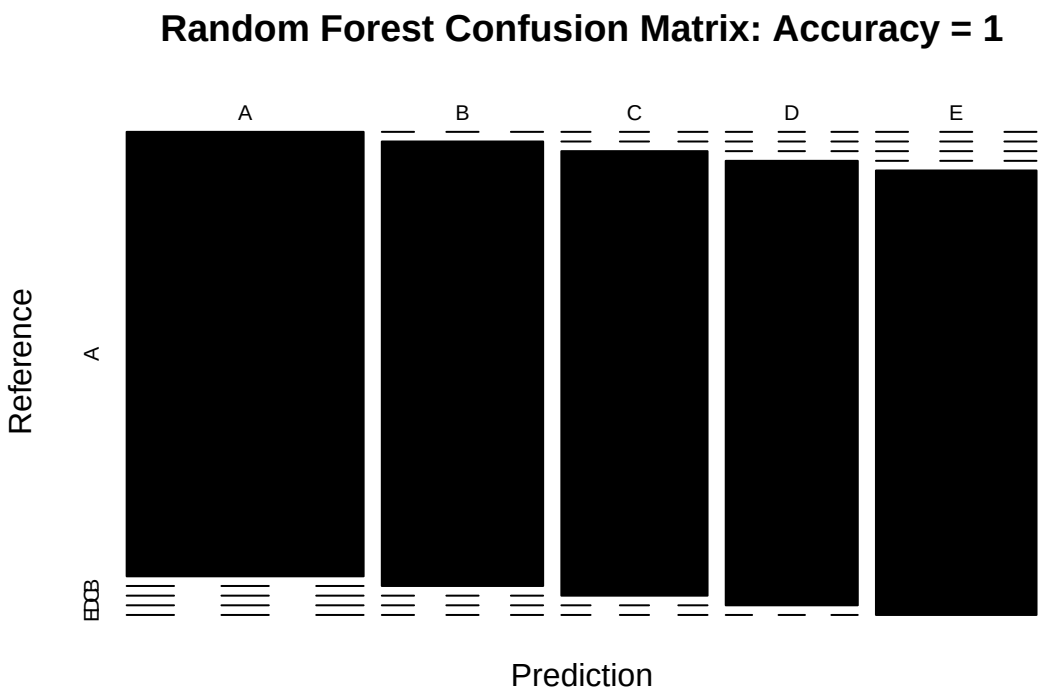
```
predictRF1 <- predict(modRF1, newdata=testData)
cmrf <- confusionMatrix(predictRF1, as.factor(testData$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1172    0    0    0    0
##          B    0  798    0    0    0
##          C    0    0  723    0    0
##          D    0    0    0  653    0
##          E    0    0    0    0  794
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2831
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Rate         0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Prevalence   0.2831   0.1928   0.1746   0.1577   0.1918
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
plot(modRF1)
```




```
plot(cmrf$table, col = cmrf$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round(cm
```




```
# 3. Generalized Boosted Regression Models

set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM  <- train(classe ~ ., data=trainData, method = "gbm", trControl = controlGBM, verbose = FALSE)
modGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

```
print(modGBM)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10990, 10990, 10989, 10991, 10988
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7485616  0.6811069
##   1                  100      0.8226676  0.7755529
##   1                  150      0.8554984  0.8171937
##   2                   50      0.8564451  0.8181992
##   2                  100      0.9055825  0.8805462
##   2                  150      0.9299697  0.9114036
##   3                   50      0.8994676  0.8727568
##   3                  100      0.9407428  0.9250337
##   3                  150      0.9608349  0.9504599
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# validate and predict

predictGBM <- predict(modGBM, newdata=testData)
cmGBM <- confusionMatrix(predictGBM, as.factor(testData$classe))
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1162   15    0    1    1
##          B    8  771   20    1    6
##          C    1   11  699   24    3
##          D    1    0    4  625    7
##          E    0    1    0    2  777
##
## Overall Statistics
##
##                Accuracy : 0.9744
##                  95% CI : (0.9691, 0.979)
##     No Information Rate : 0.2831
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9676
##
##  Mcnemar's Test P-Value : 0.000508
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9915   0.9662   0.9668   0.9571   0.9786
## Specificity            0.9943   0.9895   0.9886   0.9966   0.9991
## Pos Pred Value         0.9856   0.9566   0.9472   0.9812   0.9962
## Neg Pred Value         0.9966   0.9919   0.9929   0.9920   0.9949
## Prevalence             0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Rate         0.2807   0.1862   0.1688   0.1510   0.1877
## Detection Prevalence   0.2848   0.1947   0.1783   0.1539   0.1884
## Balanced Accuracy      0.9929   0.9778   0.9777   0.9768   0.9888
```

```
Output <- predict(modRF1, newdata=validData)
Output
```

```
##  [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
```