Practical Machine Learning assignment

Ron Collins

July 1, 2016

Practical Machine Learning/ Prediction Assignment

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, my goal is 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 (see the section on the Weight Lifting Exercise Dataset).

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Read more: http://groupware.les.inf.puc-rio.br/har#sbia_paper_section#ixzz4D5o79kNx

Goal

The goal of this 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 necessary packages

```
library(rpart)
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(corrplot)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

Data down loading and preprocessing

The training dataset will be split into training and validation datasets for development of the prediction algrothin. The test data set will be used for the project quiz.

Data down loading.

```r
# Read training data file
trainingData = read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.strin

dim(trainingData)
```

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

```r
# Read Testing data file
testingData = read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.string
dim(testingData)
```

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

split training data set into training and validation datasets.

```r
inTrain <- createDataPartition(y=trainingData$classe, p=0.7, list=FALSE)
training <- trainingData[inTrain,]
validation <- trainingData[-inTrain,]
dim(training)
```

```
## [1] 13737   160
```

```
dim(validation)
```

```
## [1] 5885  160
```

Pre-screening of the training dataset

```
From visual inspection of the training dataset it was obvious that a lot of the variables contained mis
```

```
trainingDataScreened <- training[ , colSums(is.na(trainingData)) == 0]

dim(trainingDataScreened)
```

```
## [1] 13737    60
```

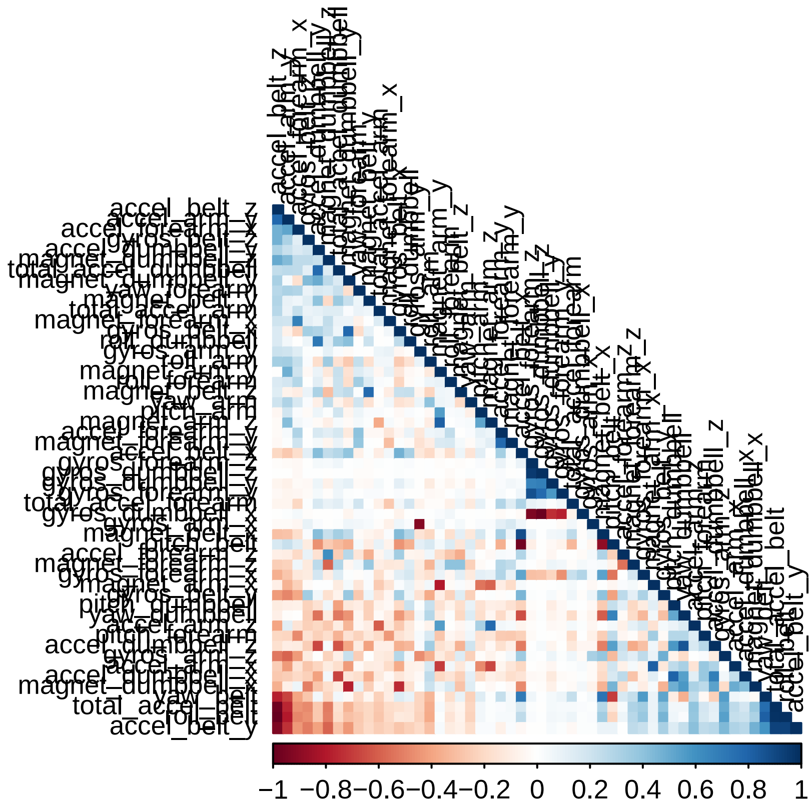
From visual inspection of the training dataset it was obvious that the following variables (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window) would not be used in development of the prediction equations. Consequently, they will be removed. This cleaning decreases the number of variables from 60 to 53.

```
remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_win
trainingDataRelevent <- trainingDataScreened[, -which(names(trainingDataScreened) %in% remove)]
dim(trainingDataRelevent)
```

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

Variables which are highly corolated(90%) will be removed using findCorrelation(). This cleaning decreases the number of variables from to 46.

```
corMatrix <- cor(trainingDataRelevent[, -53])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```




```r
corrMatrix <- cor(na.omit(trainingDataRelevent[sapply(trainingDataRelevent, is.numeric)]))
dim(corrMatrix)
```

```
## [1] 52 52
```

```r
# Remove Predictors with high correlations-90
removecor = findCorrelation(corrMatrix, cutoff = .90, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.269 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.249 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.232 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.158 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.939
##   Means:  0.103 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.952
##   Means:  0.084 vs 0.165 so flagging column 33
## All correlations <= 0.9
```

```r
trainingDataCorr = trainingDataRelevent[,-removecor]
dim(trainingDataCorr)
```

```
## [1] 13737    46
```

Development of Prediction Algrothim

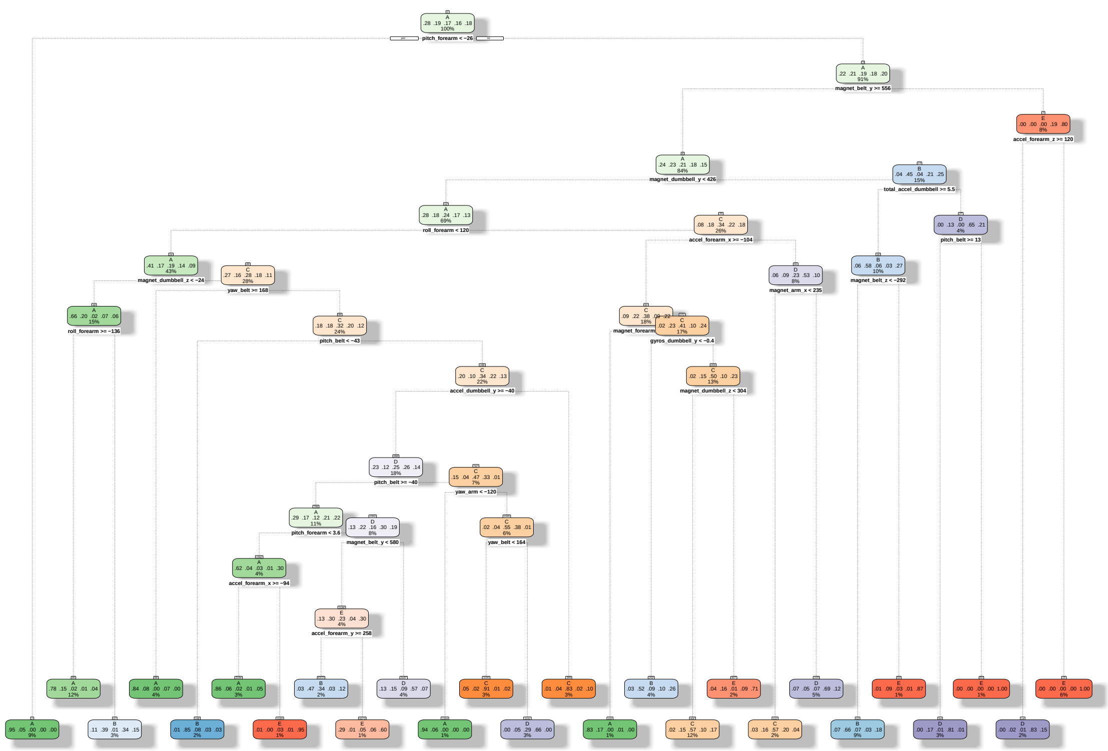
Three methods will be applied to model the regressions (in the trainingDataCorr dataset). The three models will be cross validated on the validation dataset. The models applied to the Test dataset. The model with the highest accuracy will be used for the quiz predictions. The methods are: Decision Tree, Random Forest and Generalized Boosted Model.

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

Decision Tree prediction model development

```
set.seed(11111)
fitModel<- rpart(classe ~ ., data=trainingDataCorr, method="class")
fancyRpartPlot(fitModel)
```

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



Rattle 2016−Jul−01 11:33:15 RTC

Cross Validation of the decision tree prediction using confusion Matrix

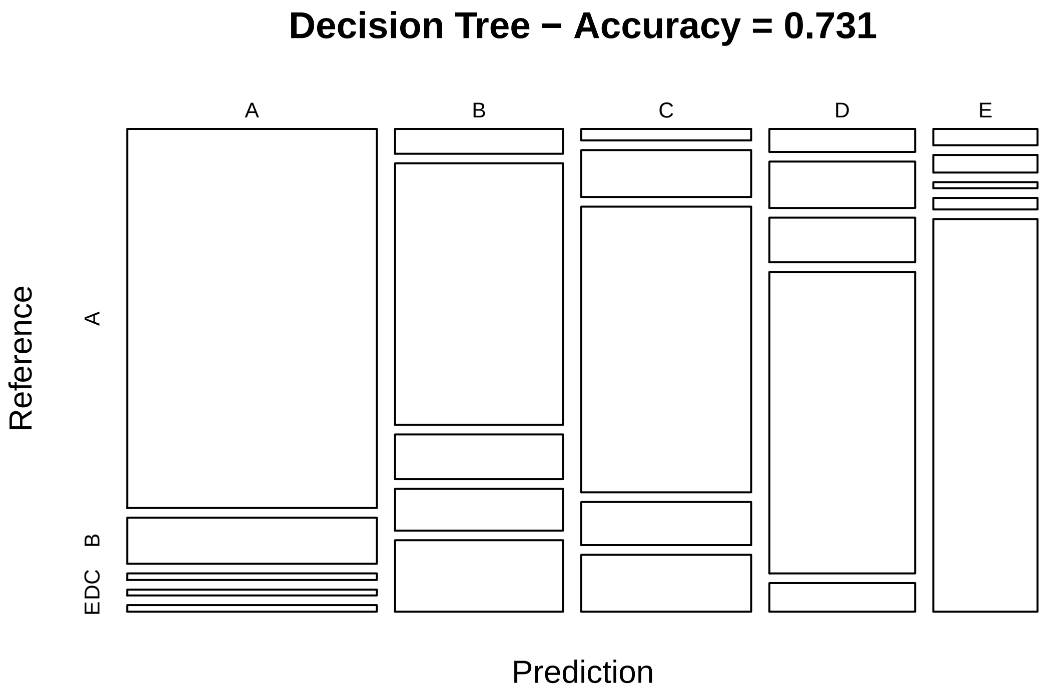
```
treePred <- predict(fitModel, newdata=validation, type="class")
confMatPredTree<- confusionMatrix(treePred, validation$classe)
confMatPredTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1497  182   26   23   26
##          B   66  695  119  111  190
##          C   31  126  768  116  153
##          D   53  107  103  695   66
##          E   27   29   10   19  647
##
## Overall Statistics
##
##                Accuracy : 0.731
##                  95% CI : (0.7195, 0.7423)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6594
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8943   0.6102   0.7485   0.7210   0.5980
## Specificity            0.9390   0.8976   0.9123   0.9331   0.9823
## Pos Pred Value         0.8535   0.5885   0.6432   0.6787   0.8839
## Neg Pred Value         0.9572   0.9056   0.9450   0.9447   0.9156
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2544   0.1181   0.1305   0.1181   0.1099
## Detection Prevalence   0.2980   0.2007   0.2029   0.1740   0.1244
## Balanced Accuracy      0.9166   0.7539   0.8304   0.8270   0.7901
```

The accuracy of the Decision Tree predictor is 0.7098.

Plot of Decision Tree predictor

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



Random Forest

```
set.seed(11111)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=trainingDataCorr, method="rf",trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 23
##
##         OOB estimate of  error rate: 0.74%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    4    1    0    2 0.001792115
## B   21 2626    8    0    3 0.012039127
## C    0   19 2369    8    0 0.011268781
## D    0    0   24 2227    1 0.011101243
## E    0    0    4    7 2514 0.004356436
```

Cross Validation od the decission tree prediction using confusion Matrix

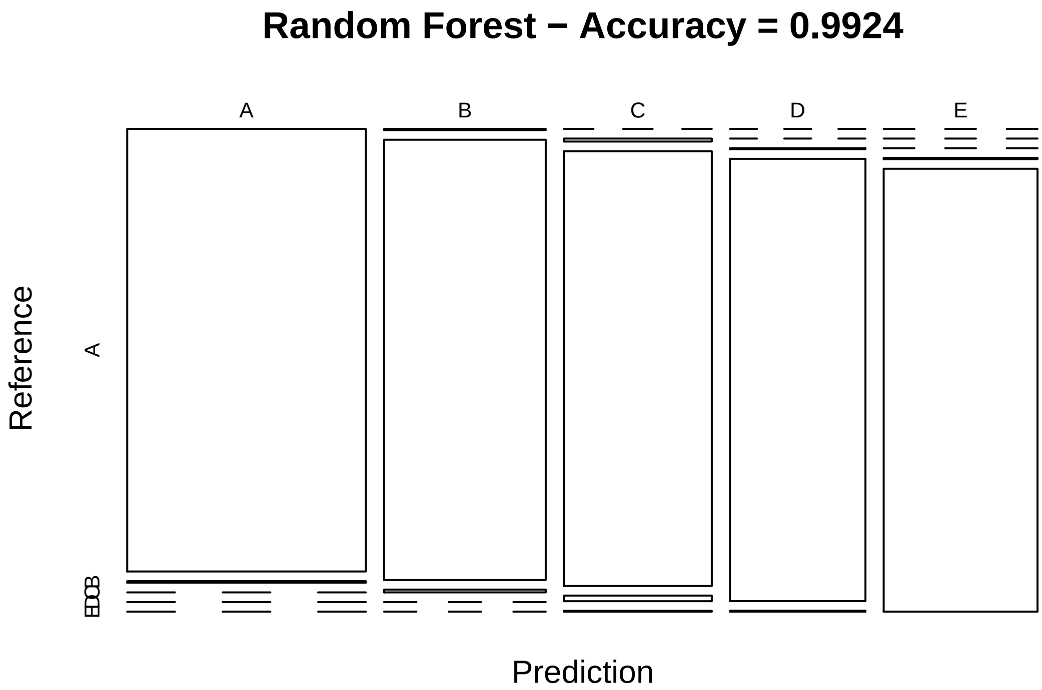
```r
predictRandForest <- predict(modFitRandForest, newdata=validation)
crossValidRandForest <- confusionMatrix(predictRandForest, validation$classe)
crossValidRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    0    0    0
##          B    3 1126    7    0    0
##          C    0    7 1017   13    2
##          D    0    0    2  948    2
##          E    0    0    0    3 1078
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9898, 0.9944)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9886   0.9912   0.9834   0.9963
## Specificity            0.9986   0.9979   0.9955   0.9992   0.9994
## Pos Pred Value         0.9964   0.9912   0.9788   0.9958   0.9972
## Neg Pred Value         0.9993   0.9973   0.9981   0.9968   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1913   0.1728   0.1611   0.1832
## Detection Prevalence   0.2850   0.1930   0.1766   0.1618   0.1837
## Balanced Accuracy      0.9984   0.9932   0.9934   0.9913   0.9978
```

The accuracy of the Random Forest model is 0.9939

Plot of Random Forest model predictors

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



Generalized Boosted Model

```
set.seed(11111)
gbmControl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
gbmModelfit  <- train(classe ~ ., data=trainingDataCorr, method = "gbm",trControl = gbmControl, verbose
```

```
## 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
```

```r
gbmModelfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 45 predictors of which 39 had non-zero influence.
```

Prediction on Validation dataset

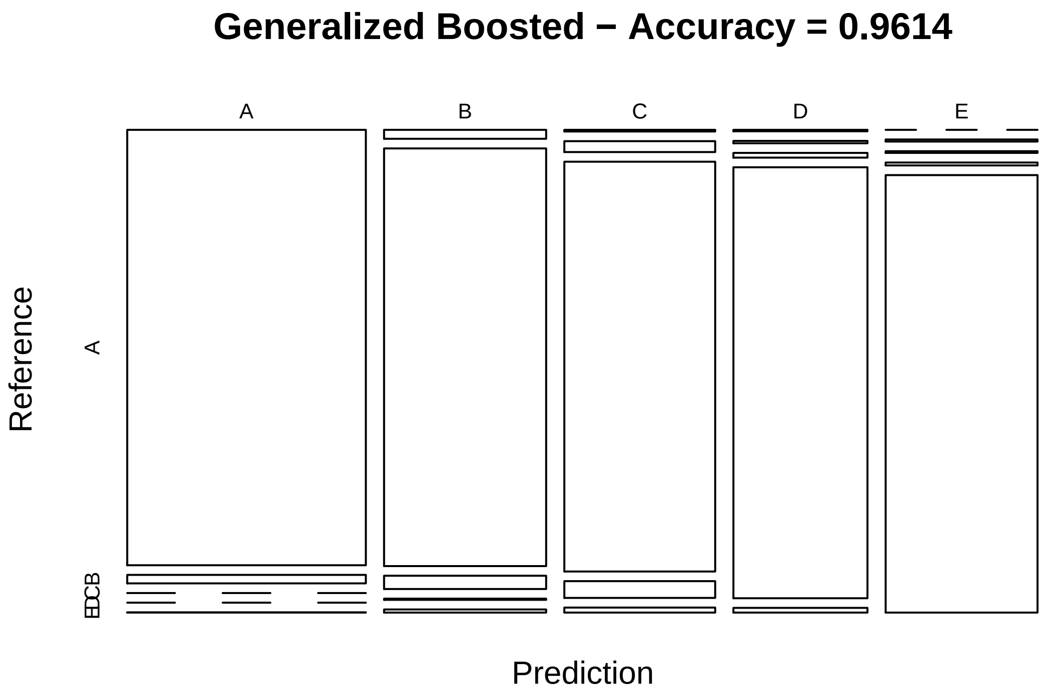
```r
predictGBM <- predict(gbmModelfit, newdata=validation)
confMatGBM <- confusionMatrix(predictGBM, validation$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1644   32    0    0    1
##          B   23 1071   34    3    8
##          C    4   26  978   40   12
##          D    3    5   10  914   10
##          E    0    5    4    7 1051
##
## Overall Statistics
##
##                Accuracy : 0.9614
##                  95% CI : (0.9562, 0.9662)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9512
##  Mcnemar's Test P-Value : 0.0001668
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9821   0.9403   0.9532   0.9481   0.9713
## Specificity            0.9922   0.9857   0.9831   0.9943   0.9967
## Pos Pred Value         0.9803   0.9403   0.9226   0.9703   0.9850
## Neg Pred Value         0.9929   0.9857   0.9901   0.9899   0.9936
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2794   0.1820   0.1662   0.1553   0.1786
## Detection Prevalence   0.2850   0.1935   0.1801   0.1601   0.1813
## Balanced Accuracy      0.9871   0.9630   0.9682   0.9712   0.9840
```

The accuracy of the Generalized Bosted Model is 0.9601

Plot of Generalized Bosted Model predictors

```r
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("Generalized Boosted - Accuracy =",
                  round(confMatGBM$overall['Accuracy'], 4)))
```



Conducting the quiz test for the fit of the selected model.

The model with the highest accuracy is the Random Forest Model.

```
Decision Tree : 0.7098
Random Forest : 0.9939
          GBM : 0.9601
```

The Random Forest model will be used to predict the results from the testing dataset.

```r
predictTEST <- predict(modFitRandForest, newdata=testingData)
predictTEST
```

```
##  [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
```