Assignment

Berj Dekramanjian

2022-10-19

Practical Machine Learning Assignment

The goal of this markdown will be to use data from accelerators on the belt, forearm, arm, and dumbbell of 6 participants to predict the manner in which they did the exercise. The report describes how the model was built, how it was cross validated, and why you different choices were made.

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.2.1
```

```
library(lattice)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.1
```

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

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

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

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.2.1
```

```
## 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.

##
## Attaching package: 'rattle'

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

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tidyr   1.2.0     v dplyr   1.0.9
## v readr   2.1.2     v stringr 1.4.1
## v purrr   0.3.4     v forcats 0.5.1

## Warning: package 'stringr' was built under R version 4.2.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine()       masks randomForest::combine()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x randomForest::margin() masks ggplot2::margin()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.1

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

Nine different packages are loaded

```
data_train <- read.csv("pml-training.csv")[,-1]
data_quiz <- read.csv("pml-testing.csv")[,-1]
dim(data_train)
```

```
## [1] 19622   159
```

```
dim(data_quiz)
```

```
## [1]  20 159
```

Both sets of data were loaded, and the dimensions of the training and testing data checked.

```
NZV <- nearZeroVar(data_train)
data_train <- data_train[, -NZV]
data_quiz <- data_quiz[, -NZV]

NaValues <- sapply(data_train, function(x) mean(is.na(x))) > 0.9
data_train <- data_train[, NaValues == "FALSE"]
data_quiz <- data_quiz[, NaValues == "FALSE"]

data_train <- data_train[,-c(1:5)]
data_quiz <- data_quiz[,-c(1:5)]

dim(data_train)
```

```
## [1] 19622    53
```

```
dim(data_quiz)
```

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

The data was cleaned by: -first removing any predictors that have missin or non-unique values -then removing any cases that have missing values -then the id and time variables were removed -finaly the dimensions of the datasets was checked again

```
in_train  <- createDataPartition(data_train$classe, p=0.75, list=FALSE)
train_set <- data_train[ in_train, ]
test_set  <- data_train[-in_train, ]

dim(train_set)
```

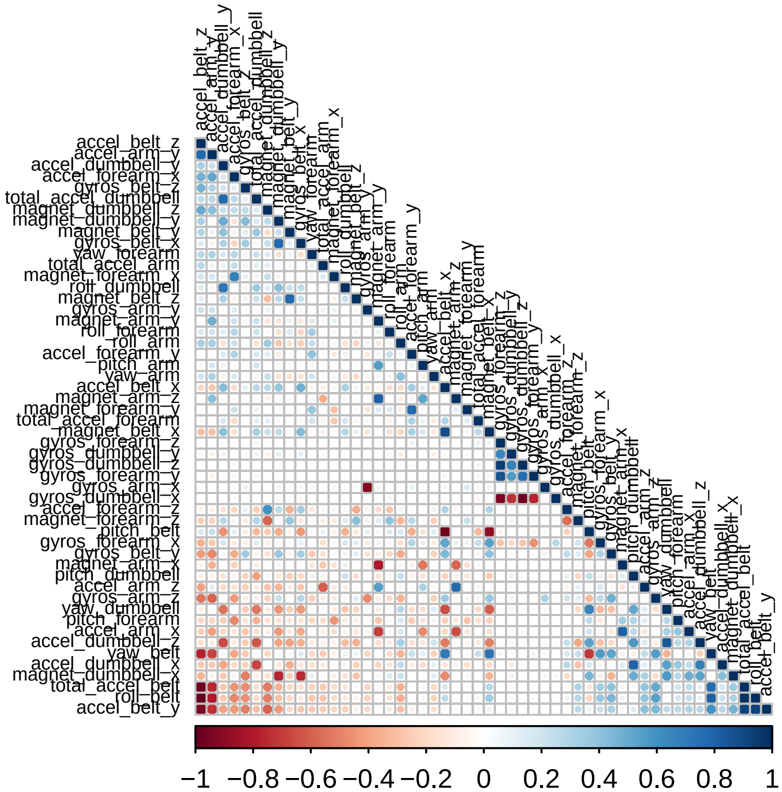
```
## [1] 14718    53
```

```
dim(test_set)
```

```
## [1] 4904   53
```

test data was partitioned for further analysis

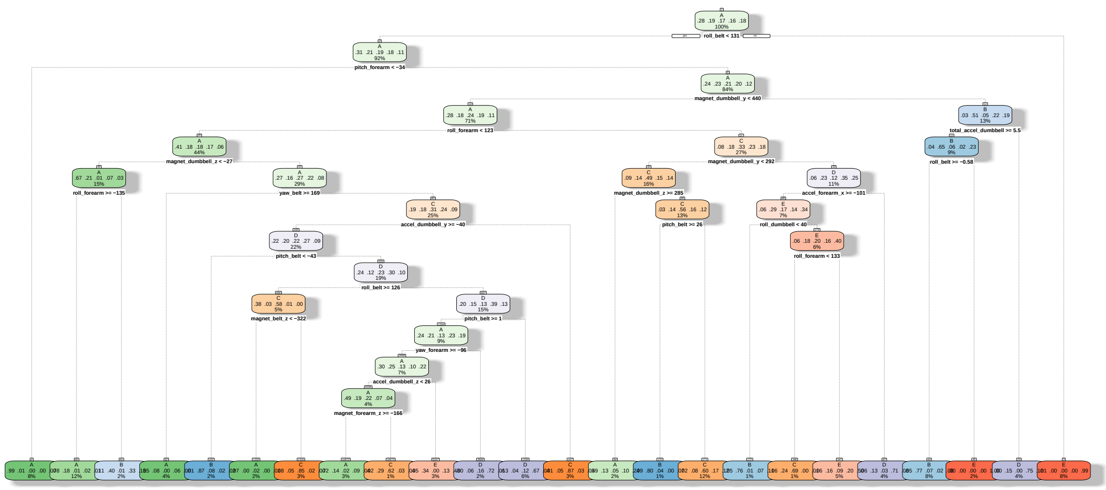
```
corr_matrix <- cor(train_set[ ,-53])
corrplot(corr_matrix, order = "FPC", method = "circle", type = "lower",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



Since there aren't that many variables that are correlated, it seems multiple prediction models might be needed. first off with a decision tree.

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

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



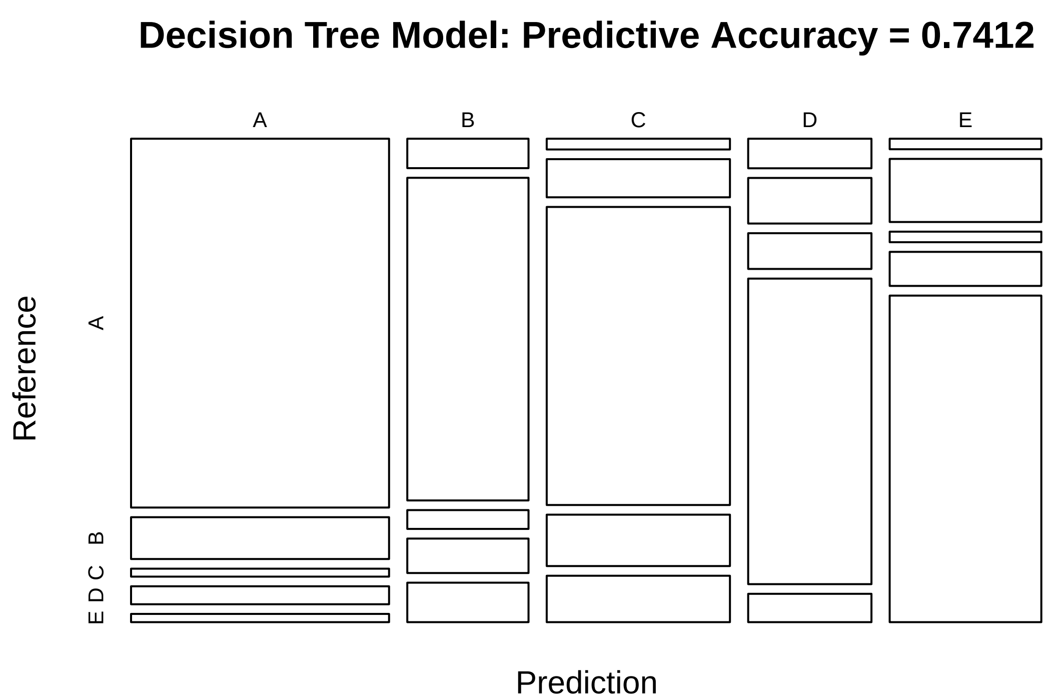
Rattle 2022−Oct−20 16:42:20 u176055

```r
predict_decision_tree <- predict(fit_decision_tree, newdata = test_set, type="class")
conf_matrix_decision_tree <- confusionMatrix(predict_decision_tree, factor(test_set$classe))
conf_matrix_decision_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1253  142   27   61   28
##          B   47  515   30   55   63
##          C   26   92  719  124  112
##          D   48   74   58  496   46
##          E   21  126   21   68  652
##
## Overall Statistics
##
##                Accuracy : 0.7412
##                  95% CI : (0.7287, 0.7534)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6719
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8982   0.5427   0.8409   0.6169   0.7236
## Specificity            0.9265   0.9507   0.9126   0.9449   0.9410
## Pos Pred Value         0.8293   0.7254   0.6701   0.6870   0.7342
## Neg Pred Value         0.9581   0.8965   0.9645   0.9264   0.9380
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2555   0.1050   0.1466   0.1011   0.1330
## Detection Prevalence   0.3081   0.1448   0.2188   0.1472   0.1811
## Balanced Accuracy      0.9123   0.7467   0.8768   0.7809   0.8323
```

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



The decision trees predictive accuracey was relatively low at 73.5 percent. next up we try the generalized boosted model.

```
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 52 predictors of which 52 had non-zero influence.
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1367   19    0    0    2
##          B   18  894   24    1    9
##          C    4   31  820   32   11
##          D    4    0   10  769    8
##          E    2    5    1    2  871
##
## Overall Statistics
##
##                Accuracy : 0.9627
##                  95% CI : (0.957, 0.9678)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9528
##
##  Mcnemar's Test P-Value : 0.0001509
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9799   0.9420   0.9591   0.9565   0.9667
## Specificity            0.9940   0.9869   0.9807   0.9946   0.9975
## Pos Pred Value         0.9849   0.9450   0.9131   0.9722   0.9886
## Neg Pred Value         0.9920   0.9861   0.9913   0.9915   0.9925
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2788   0.1823   0.1672   0.1568   0.1776
## Detection Prevalence   0.2830   0.1929   0.1831   0.1613   0.1796
## Balanced Accuracy      0.9870   0.9644   0.9699   0.9756   0.9821
```

GBM did quite well with a better accuracy of 96.6 percent

Lastly we do a random forest model

```r
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: 0.72%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4178    6    0    0    1 0.001672640
## B   23 2816    6    1    2 0.011235955
## C    0   13 2544   10    0 0.008959875
## D    0    1   29 2379    3 0.013681592
## E    0    1    3    7 2695 0.004065041
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    2    0    0    0
##          B    3  943    6    1    0
##          C    0    4  844    4    2
##          D    0    0    5  798    3
##          E    0    0    0    1  896
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.991, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9937   0.9871   0.9925   0.9945
## Specificity            0.9994   0.9975   0.9975   0.9980   0.9998
## Pos Pred Value         0.9986   0.9895   0.9883   0.9901   0.9989
## Neg Pred Value         0.9991   0.9985   0.9973   0.9985   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1923   0.1721   0.1627   0.1827
## Detection Prevalence   0.2843   0.1943   0.1741   0.1644   0.1829
## Balanced Accuracy      0.9986   0.9956   0.9923   0.9953   0.9971
```

Predictive accuracy of the Random Forest model is even better at 99.4 percent

we are going to go ahead and use the random forest model for our predicitons for the quiz.

```r
predict_quiz <- as.data.frame(predict(fit_RF, newdata = data_quiz))
predict_quiz
```

```
##    predict(fit_RF, newdata = data_quiz)
```

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