Taisekwa Prediction assingment

Taisekwa Chikazhe

17 September 2021

Overview

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, we will 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.

The data consists of a Training data and a Test data (to be used to validate the selected model).

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.

Loading required packages

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

library(corrplot)

## corrplot 0.90 loaded

library(gbm)

## Loaded gbm 2.1.8

library(readr)
library(lattice)
library(tidyverse)

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

## v ggplot2 3.3.5     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.1

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

library(caret)

##
## Attaching package: 'caret'

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

library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

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

```
library(rattle)
library(corrplot)
```

Load the required data

```
traincsv <-
read_csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv")

## New names:
## * `` -> ...1

## Rows: 19622 Columns: 160

## -- Column specification ----------------------------------------------
------
## Delimiter: ","
## chr  (34): user_name, cvtd_timestamp, new_window, kurtosis_roll_belt,
kurtos...
## dbl (126): ...1, raw_timestamp_part_1, raw_timestamp_part_2, num_window,
rol...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.

testcsv <- read_csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv")

## New names:
## * `` -> ...1

## Rows: 20 Columns: 160

## -- Column specification ----------------------------------------------
------
## Delimiter: ","
## chr   (3): user_name, cvtd_timestamp, new_window
## dbl  (57): ...1, raw_timestamp_part_1, raw_timestamp_part_2, num_window,
rol...
## lgl (100): kurtosis_roll_belt, kurtosis_picth_belt, kurtosis_yaw_belt,
skewn...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

check for data dimesions

```
dim(traincsv)
```

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

```
dim(testcsv)
```

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

Data cleaning

```
traincsv <- traincsv[,colMeans(is.na(traincsv)) < .9] #removing mostly na columns
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
traincsv <- traincsv[,-c(1:7)] #removing metadata which is irrelevant to the outcome
dim(traincsv)
```

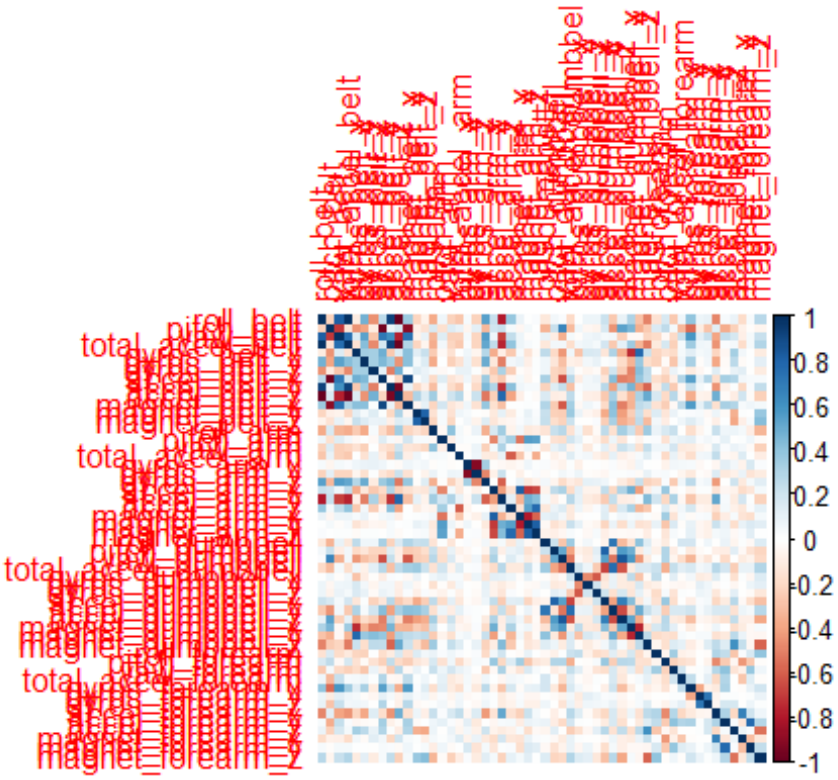
```
## [1] 19622    53
```

Partitioning data, into training and testing data set

```
inTrain <- createDataPartition(y=traincsv$classe, p=0.7, list=F)
train <- traincsv[inTrain,]
valid <- traincsv[-inTrain,]
```

Ploting correlation plot

```
corrPlot <- cor(train[, -length(names(train))])
corrplot(corrPlot, method="color")
```



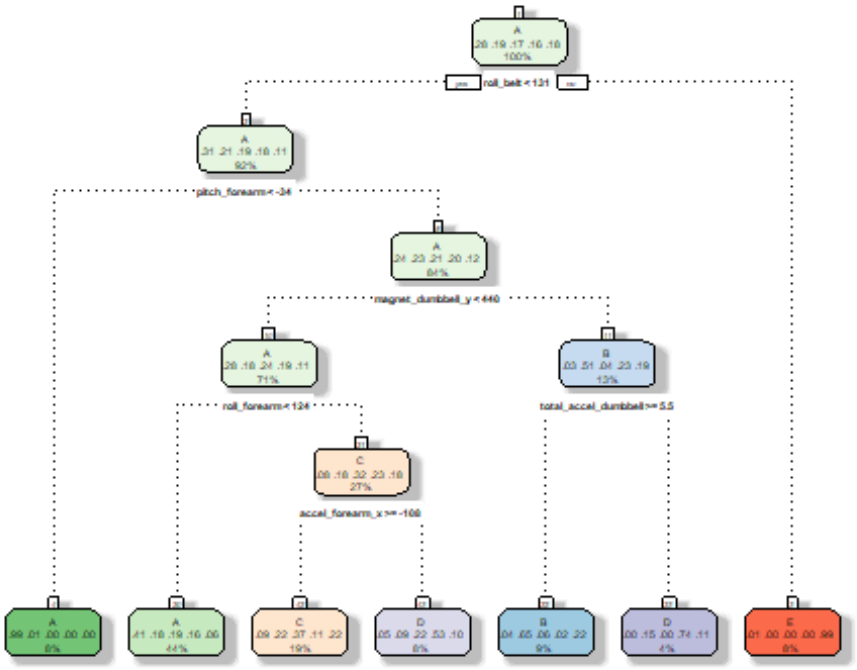
Creating and testing models Here we will test a few popular models including: Decision Trees, Random Forest, Gradient Boosted Trees, and SVM.

```
#Set up control for training to use 3-fold cross validation.
control <- trainControl(method="cv", number=3, verboseIter=F)
```

Decision tree model

```
mod_trees <- train(classe~., data=train, method="rpart", trControl = control,
tuneLength = 5)

fancyRpartPlot(mod_trees$finalModel)
```



Rattle 2021-Sep-20 10:24:40 ChikazheT

#Prediction

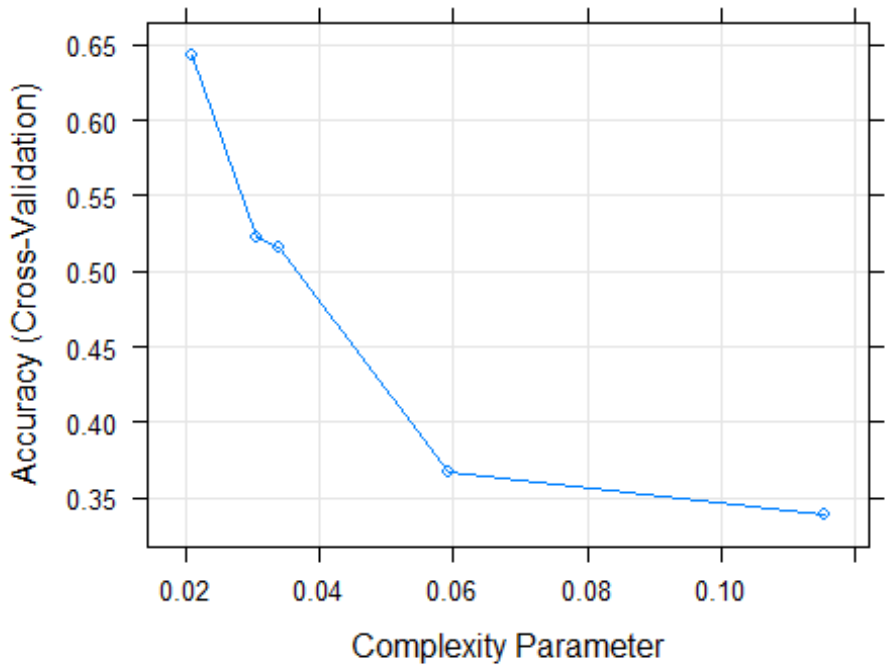
```
pred_trees <- predict(mod_trees, valid)
cmtrees <- confusionMatrix(pred_trees, factor(valid$classe))
cmtrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1529  490  469  456  162
##          B   25  361   31    9  130
##          C   86  221  422  119  233
##          D   29   67  104  380   70
##          E    5    0    0    0  487
##
## Overall Statistics
##
##                Accuracy : 0.5402
##                  95% CI : (0.5274, 0.553)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3998
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9134  0.31694  0.41131  0.39419  0.45009
## Specificity            0.6255  0.95891  0.86438  0.94513  0.99896
## Pos Pred Value         0.4923  0.64928  0.39038  0.58462  0.98984
## Neg Pred Value         0.9478  0.85401  0.87427  0.88844  0.88967
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2598  0.06134  0.07171  0.06457  0.08275
## Detection Prevalence   0.5278  0.09448  0.18369  0.11045  0.08360
## Balanced Accuracy      0.7694  0.63793  0.63784  0.66966  0.72453
```

Decision tree cross validation

```
plot(mod_trees)
```



Random forest model

```
mod_rf <- train(classe~., data=train, method="rf", trControl = control,
tuneLength = 5)
```

Random forest prediction

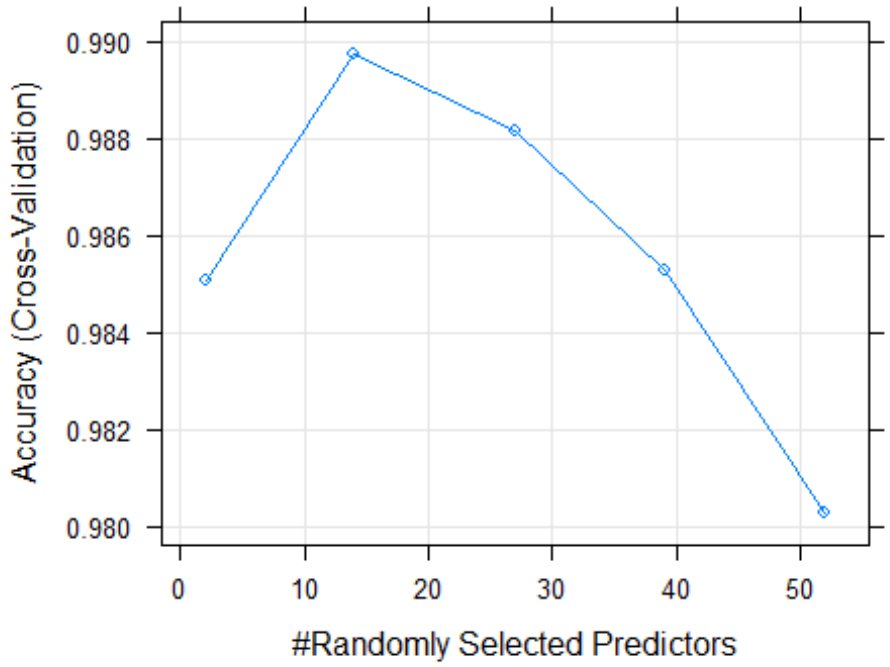
```
pred_rf <- predict(mod_rf, valid)
cmrf <- confusionMatrix(pred_rf, factor(valid$classe))
cmrf

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    4    0    0    0
##          B    2 1134    5    0    0
##          C    0    1 1020    6    1
##          D    0    0    1  950    6
##          E    1    0    0    8 1075
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9956   0.9942   0.9855   0.9935
## Specificity            0.9991   0.9985   0.9984   0.9986   0.9981
## Pos Pred Value         0.9976   0.9939   0.9922   0.9927   0.9917
## Neg Pred Value         0.9993   0.9989   0.9988   0.9972   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1927   0.1733   0.1614   0.1827
## Detection Prevalence   0.2846   0.1939   0.1747   0.1626   0.1842
## Balanced Accuracy      0.9986   0.9971   0.9963   0.9920   0.9958
```

Random forest cross validation

```
plot(mod_rf)
```



Gradient boosted trees model

```
mod_gbm <- train(classe~., data=train, method="gbm", trControl = control,
tuneLength = 5, verbose = F)
```

Gradient boosted trees model prediction

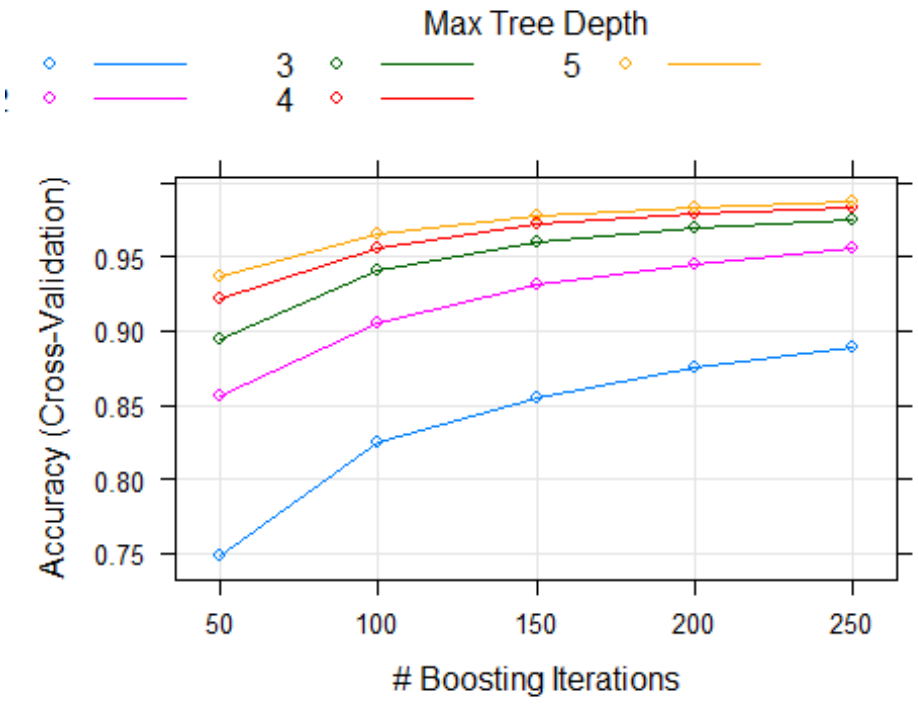
```
pred_gbm <- predict(mod_gbm, valid)
cmgbm <- confusionMatrix(pred_gbm, factor(valid$classe))
cmgbm

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1666    9    0    0    0
##          B    6 1125    5    0    1
##          C    1    5 1014   13    2
##          D    0    0    7  942    5
##          E    1    0    0    9 1074
##
## Overall Statistics
##
##                Accuracy : 0.9891
##                  95% CI : (0.9861, 0.9916)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9862
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9877   0.9883   0.9772   0.9926
## Specificity            0.9979   0.9975   0.9957   0.9976   0.9979
## Pos Pred Value         0.9946   0.9894   0.9797   0.9874   0.9908
## Neg Pred Value         0.9981   0.9971   0.9975   0.9955   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2831   0.1912   0.1723   0.1601   0.1825
## Detection Prevalence   0.2846   0.1932   0.1759   0.1621   0.1842
## Balanced Accuracy      0.9965   0.9926   0.9920   0.9874   0.9953
```

GBM cross validation

```
plot(mod_gbm)
```



Support vector machine model

```
mod_svm <- train(classe~., data=train, method="svmLinear", trControl =
control, tuneLength = 5, verbose = F)
```

Model prediction

```
pred_svm <- predict(mod_svm, valid)
cmsvm <- confusionMatrix(pred_svm, factor(valid$classe))
cmsvm

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1535  157   99   82   61
##          B   36  806   73   40  146
##          C   45   75  800   99   55
##          D   48   15   29  706   54
##          E   10   86   25   37  766
##
## Overall Statistics
##
##                Accuracy : 0.7839
##                  95% CI : (0.7731, 0.7943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.725
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9170   0.7076   0.7797   0.7324   0.7079
## Specificity            0.9052   0.9378   0.9436   0.9703   0.9671
## Pos Pred Value         0.7937   0.7321   0.7449   0.8286   0.8290
## Neg Pred Value         0.9648   0.9304   0.9530   0.9487   0.9363
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2608   0.1370   0.1359   0.1200   0.1302
## Detection Prevalence   0.3286   0.1871   0.1825   0.1448   0.1570
## Balanced Accuracy      0.9111   0.8227   0.8617   0.8513   0.8375
```

Results (Accuracy & Out of Sample Error)

```
 accuracy oos_error
```

Tree 0.537 0.463 RF 0.996 0.004 GBM 0.992 0.008 SVM 0.781 0.219 The best model is the Random Forest model, with 0.9957519 accuracy and 0.0042481 out of sample error rate.