Machine Learning Course Project

Manu

31/10/2020

```r
rm(list=ls())
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(ggplot2)
library(rattle)
library(AppliedPredictiveModeling)
library(e1071)
library(dplyr)

# Set seed for reproducibility
set.seed(1234)
```

1. Introduction

1.1 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, your goal will 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).

1.2 Data

- 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.

1.3 Explanation

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions. The goal of the project is to predict the manner in which they did the exercise.

1.4 Strategy

1.4.1 Strategy for building the model

The outcome variable is a factor variable containing five classes:

- A: exactly according to the specification
- B: throwing elbows to the front
- C: lifting the dumbbel only halfway
- D: lowering the dumbbell only halfway
- E: throwing the hips to the front

The goal, on the base of all other variables in this dataset, is to predict with maximum accuracy and minimum out-of-sample error, the way in which the exercice was done.

1.4.2 Strategy for crossvalidation and cleaning of the data

The cross validation was done by further dividing the training data set into two subsamples:

- subtraining data : 75% (from the training data)
- subtesting : 25% (from the training data)

At the level of features, the predictors with near zero variance and non-relevant for the prediction models were removed.

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

# Clean data:
        # 1) By removing zero covariates
dim(training)
```

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

```
nzvTrain <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[, !nzvTrain$nzv]
dim(training)
```

```
## [1] 19622   100
```

```
        # 2) By removing irrelevant vars
training <- training[,-c(1:7)]
dim(training)
```

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

```
        # 3) By removing var with NA's
                # This it not optimal, it was done to reduce computational time
training <- training[,colSums(is.na(training))== 0]

# Partition training into subsets
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subtraining <- training[inTrain, ]
subtesting <- training[-inTrain, ]

dim(training)
```

```
## [1] 19622    52
```

```
dim(subtraining)
```

```
## [1] 14718    52
```

```
dim(subtesting)
```

```
## [1] 4904   52
```

```
rm(inTrain, nzvTrain)
```

1.4.3 Expected out-of-sample error

The out-of-sample error is the rate reported on the data not used to build the model. In the present case, it will be done in two steps. The first step will be building the model on the subtraining set and testing the model on the subtesting set. The second step will be applying the model on the main testing set.

The expected out-of-sample errors is the number of missclassified observation / total observations in the subtesting set.

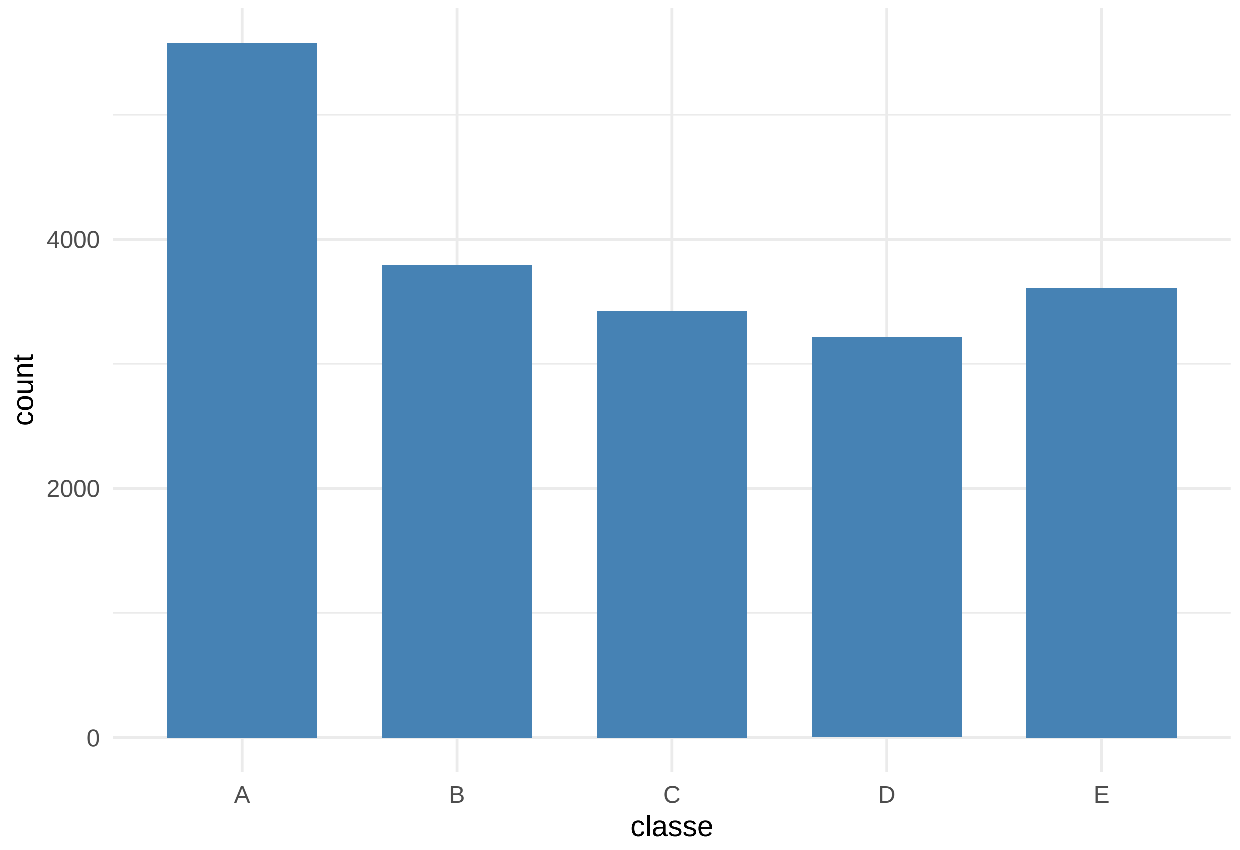
1.4.4 Reproducibility

A pseudo-rendom number generator was set to 1234.

2. Prediction models

The outcome variable classe has a certain balance between its different categories which good. The category A is the most frequent.

```
ggplot(training, aes(x=classe)) +
        geom_bar(stat="count", width=0.7, fill="steelblue") +
        theme_minimal()
```



Five different models were built and their accuracy compared :

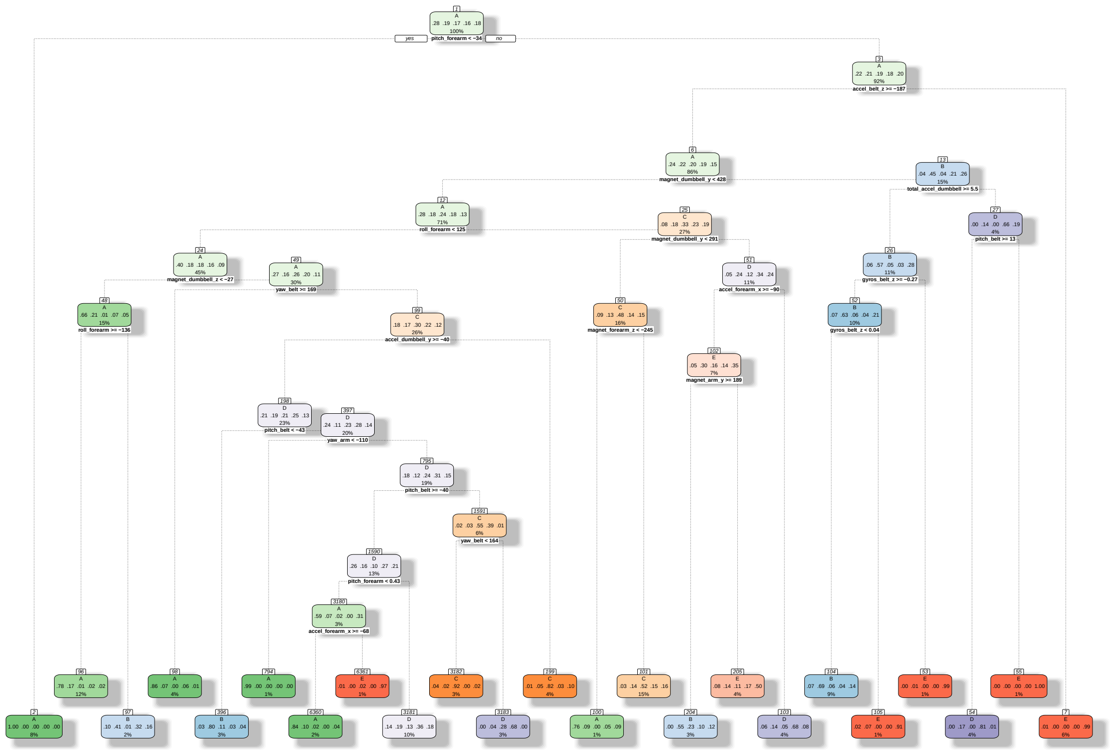
- Decision Tree
- Random Forest
- Boosting
- Support Vector Machine
- Ensembling all previous models without decision tree
- Ensembling all previous models

Computational time was high, especially in the case of Random Forest. To avoid doing twice the same computational runs, the models were saved and simply called back when needed to make new predictions. This is the reason why the models are commented underneath, to avoid having to run them again at every knit. A comparison table of accuracy is shown at the end of this section.

2.1 Decision Tree

```r
# modFitDT <- rpart(classe ~ ., data=subtraining, method="class")

# saveRDS(modFitDT, "./models/modFitDT.rds")

# Load model
modFitDT1 <- readRDS("./models/modFitDT.rds")

predictionDT <- predict(modFitDT1, newdata=subtesting, type="class")

cfmDT <- confusionMatrix(subtesting$classe, predictionDT)
AccuracyDecisionTree <- cfmDT$overall['Accuracy']
```

```
AccuracyDecisionTreeCILo <- cfmDT$overall['AccuracyLower']
AccuracyDecisionTreeCIUp <- cfmDT$overall['AccuracyUpper']


fancyRpartPlot(modFitDT1)
```



Rattle 2020−Nov−02 06:46:39 clivaz

2.2 Random Forest

```
## Use na.roughfix to imputes missing values by median/mode

# modFitRF <- train(classe ~ ., data=subtraining, method="rf", prox=TRUE, na.action=na.roughfix)

# saveRDS(modFitRF, "./models/modFitRF.rds")

# Load model
modFitRF1 <- readRDS("./models/modFitRF.rds")

predictionRF <- predict(modFitRF1, newdata=subtesting)
cfmRF <- confusionMatrix(subtesting$classe, predictionRF)
AccuracyRandomForest <- cfmRF$overall['Accuracy']
AccuracyRandomForestCILo <- cfmRF$overall['AccuracyLower']
AccuracyRandomForestCIUp <- cfmRF$overall['AccuracyUpper']
```

2.3 Boosting

```r
# modFitBoost <- train(classe ~., data=subtraining, method="gbm", verbose=FALSE)

# saveRDS(modFitBoost, "./models/modFitBoost.rds")

# Load model
modFitBoost1 <- readRDS("./models/modFitBoost.rds")

predictionBoost <- predict(modFitBoost1, newdata=subtesting)
cfmBoost <- confusionMatrix(subtesting$classe, predictionBoost)
AccuracyBoosting <- cfmBoost$overall['Accuracy']
AccuracyBoostingCILo <- cfmBoost$overall['AccuracyLower']
AccuracyBoostingCIUp <- cfmBoost$overall['AccuracyUpper']
```

2.4 Support Vector machine

```r
# modFitSupVect <- svm(classe ~ ., data=subtraining)

# saveRDS(modFitSupVect, "./models/modFitSupVect.rds")

# Load model
modFitSupVect1 <- readRDS("./models/modFitSupVect.rds")

predictionSupVect <- predict(modFitSupVect1, newdata=subtesting)
cfmSupVect <- confusionMatrix(subtesting$classe, predictionSupVect)
AccuracySupportVector <- cfmSupVect$overall['Accuracy']
AccuracySupportVectorCILo <- cfmSupVect$overall['AccuracyLower']
AccuracySupportVectorCIUp <- cfmSupVect$overall['AccuracyUpper']
```

2.5 Ensembling

```r
# Data combined
        # Ensembling on all vars
combinedPredictionDfAll <- data.frame(predictionDT, predictionRF, predictionBoost, predictionSupVect, cl
        # prediction DT not considered bec accuracy too low
combinedPredictionDf <- data.frame(predictionRF, predictionBoost, predictionSupVect, classe=subtesting$

# Model fit on data combined
# modFitEnsemblingAll <- train(classe ~ ., data=combinedPredictionDfAll, method="rf", prox=TRUE)
# modFitEnsembling <- train(classe ~ ., data=combinedPredictionDf, method="rf", prox=TRUE)

# saveRDS(modFitEnsemblingAll, "./models/modFitEnsemblingAll.rds")
# saveRDS(modFitEnsembling, "./models/modFitEnsembling.rds")

# Load model
modFitEnsemblingAll1 <- readRDS("./models/modFitEnsemblingAll.rds")
modFitEnsembling1 <- readRDS("./models/modFitEnsembling.rds")

predictionEnsemblingAll <- predict(modFitEnsemblingAll1, newdata=subtesting)
predictionEnsembling <- predict(modFitEnsembling1, newdata=subtesting)

cfmEnsAll <- confusionMatrix(subtesting$classe, predictionEnsemblingAll)
```

```r
cfmEns <- confusionMatrix(subtesting$classe, predictionEnsembling)


AccuracyEnsemblingAll <- cfmEnsAll$overall['Accuracy']
AccuracyEnsemblingAllCILo <- cfmEnsAll$overall['AccuracyLower']
AccuracyEnsemblingAllCIUp <- cfmEnsAll$overall['AccuracyUpper']

AccuracyEnsemblingNoDT <- cfmEns$overall['Accuracy']
AccuracyEnsemblingNoDTCILo <- cfmEns$overall['AccuracyLower']
AccuracyEnsemblingNoDTCIUp <- cfmEns$overall['AccuracyUpper']
```

3. Accuracy

3.1 Accuracy comparison

The following table shows that the best model is the one combining all models (Decision Tree, Random Forest, Boosting, Support Vector Machine). It has a value of 0.9955. The high value of Random Forest with 0.9945 is worth noticing. The confidence interval (95%) is written in the table as accuracy lower and upper.

```r
# Accuracy value
accuracyDf <- t(data.frame(AccuracyDecisionTree, AccuracyRandomForest, AccuracyBoosting, AccuracySuppor
accuracyDf <- as.data.frame(accuracyDf)

# Accuracy CI Lower
accuracyDfCILo <- t(data.frame(AccuracyDecisionTreeCILo, AccuracyRandomForestCILo, AccuracyBoostingCILo
accuracyDfCILo <- as.data.frame(accuracyDfCILo)

# Accuracy CI Upper
accuracyDfCIUp <- t(data.frame(AccuracyDecisionTreeCIUp, AccuracyRandomForestCIUp, AccuracyBoostingCIUp
accuracyDfCIUp <- as.data.frame(accuracyDfCIUp)

# Show accuracy table
accuracySummaryDf <- data.frame(accuracyDf, accuracyDfCILo, accuracyDfCIUp)
print(accuracySummaryDf, digits = 4)
```

```
##                          Accuracy AccuracyLower AccuracyUpper
## AccuracyDecisionTree       0.7019        0.6889        0.7147
## AccuracyRandomForest       0.9945        0.9920        0.9964
## AccuracyBoosting           0.9635        0.9579        0.9686
## AccuracySupportVector      0.9437        0.9369        0.9500
## AccuracyEnsemblingNoDT     0.9949        0.9925        0.9967
## AccuracyEnsemblingAll      0.9955        0.9932        0.9972
```

4. Submission

Here is the final resul based on the prediction model "Random Forest" applied on the testing data set.

```r
# Equalize testing columns with the features defined in the training models
        # If not, support vector machine model fail
colNames <- colnames(subtesting)
colNames <- colNames[!(colNames %in% "classe")]
testing <- testing %>%
        select(one_of(colNames))
```

```r
# Predict on testing (final) set with Random Forest
predictionRF_F <- predict(modFitRF1, newdata=testing)
predictionRF_F
```

```
##  [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
```