Final_project

Pingping

8/16/2020

Introduction

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. 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 use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The participants perform barbell lifts correctly and incorrectly in 5 different ways.

The goal of this project is to predict the manner in which they did the exercise, which is the "classe" variable in the training set. We've created a report describing how we built different models, how we used cross validation, and give the expected out of sample error. Finally, we used the prediction model to predict 20 different test cases.

Data loading and cleaning

```r
setwd("~/R.Studio/Machine_learning/Practical_Machine_Learning_Project")
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
library(gbm)
```

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. We thank the providors as they have been very generous in allowing their data to be used for this kind of assignment.

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

```r
urltrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urltest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainpml <- read.csv(url(urltrain))
testpml  <- read.csv(url(urltest))
#split the training data into train set and test set for cross validation
set.seed(12345)
inTrain <- createDataPartition(trainpml$classe, p = 0.7, list = F)
```

```
training <- trainpml[inTrain, ]
testing <- trainpml[-inTrain, ]
training$classe <- as.factor(training$classe)
testing$classe <- as.factor(testing$classe)
testing <- trainpml[-inTrain, ]
dim(training)
```

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

```
dim(testing)
```

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

```
sum(is.na.data.frame(training))
```

```
## [1] 901418
```

As we can see, the training set and testing set both have 160 variables. The training set has 13737 observations, while the testing set has 5885 observations.However, for now the variables have a lot of NAs (901753), which should be removed to get clean data. The Near Zero variance (NZV) variables and the ID variables will also be removed.

```
# remove the near zero variance variables
zerovar <- nearZeroVar(training)
training1 <- training[ ,-zerovar]
testing1 <- testing[ ,-zerovar]
testpml1 <- testpml[ ,-zerovar]
# remove the almost (>95%) NA variables
trainingNA <- is.na(training1)
training2 <- training1[ , colMeans(trainingNA)<0.95]
testing2 <- testing1[ , colMeans(trainingNA)<0.95]
testpml2 <- testpml1[ , colMeans(trainingNA)<0.95]
# remove the ID variables
training3 <- training2[ ,-(1:5)]
testing3 <- testing2[ ,-(1:5)]
testpml3 <- testpml2[ ,-(1:5)]
dim(training3)
```

```
## [1] 13737    54
```

```
dim(testing3)
```

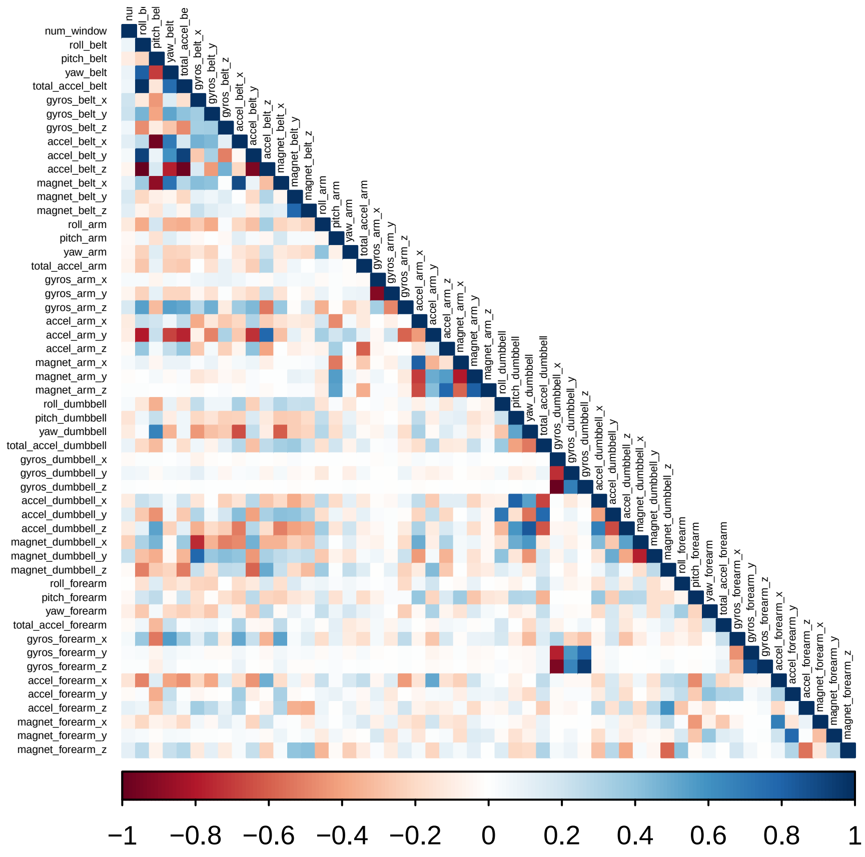
```
## [1] 5885   54
```

After proper data cleaning, now we have 54 variables left in the training and testing datasets.

Exploratory analysis

First, we'd like to have a brief idea about the correlation between different variables.

```r
corMatrix <- cor(training3[, -54])
corrplot(corMatrix, order = "original", method = "color", type = "lower", tl.cex = 0.3, tl.col = rgb(0,
```



As we can see, some of the variables are highly correlated, which are shown in dark colors. ## Modelling and Prediction

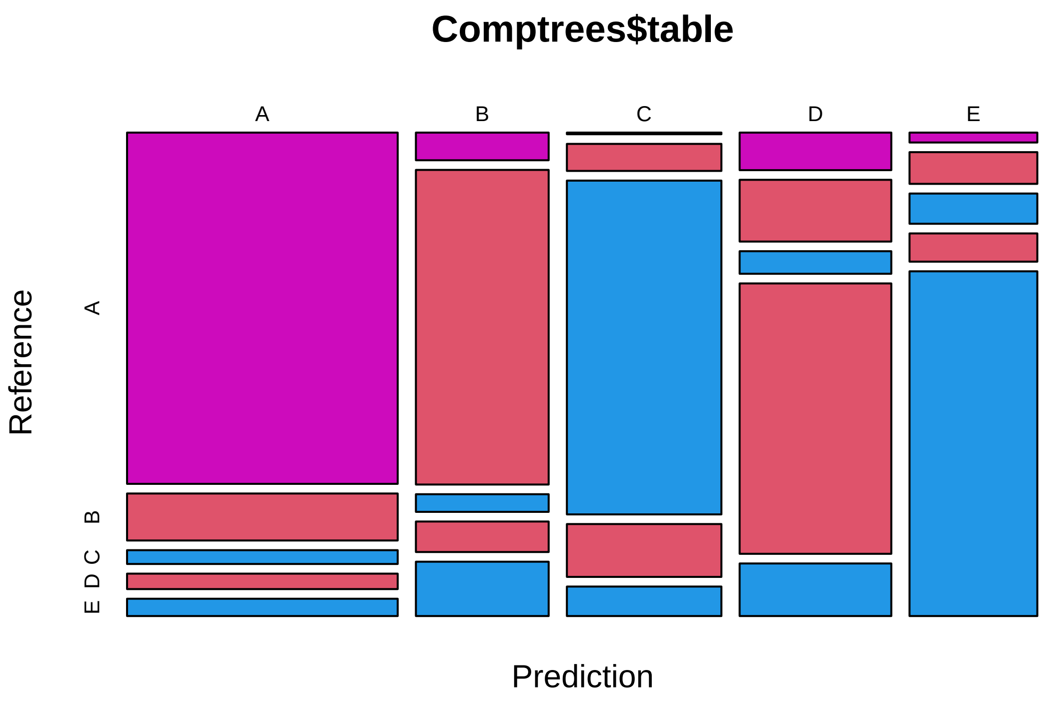
Prediction with Trees

First, we'll try prediction with trees.

```r
set.seed(12345)
training3$classe <- as.factor(training3$classe)
testing3$classe <- as.factor(testing3$classe)
modtrees <- rpart(classe~., training3)
predtrees0 <- predict(modtrees, testing3)
predtrees <- factor(colnames(predtrees0)[apply(predtrees0, 1, which.max)])
Comptrees <- confusionMatrix(predtrees, testing3$classe)
Comptrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  201   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## Overall Statistics
##
##                Accuracy : 0.7342
##                  95% CI : (0.7228, 0.7455)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973   0.5795   0.7943   0.6722   0.6433
## Specificity            0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value         0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value         0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence   0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy      0.9011   0.7610   0.8693   0.7936   0.8006
```

```r
plot(Comptrees$table, col = Comptrees$table)
```



As we can see, the prediction with trees method has an accuracy of 0.7342.

Random Forest

We will then use the Random Forest method.

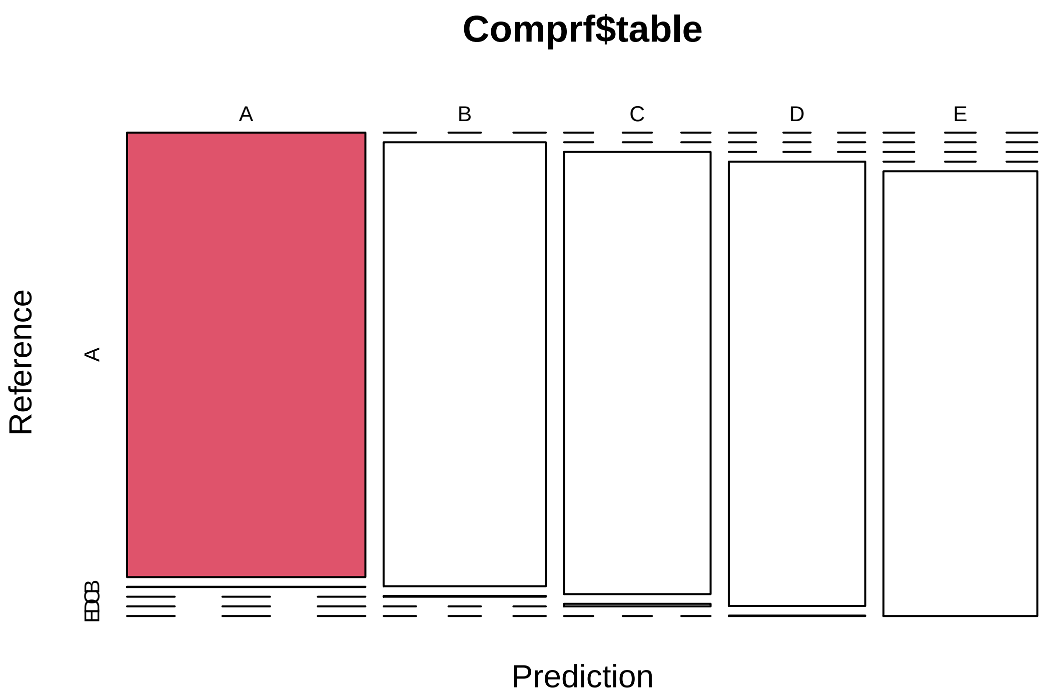
```r
library(ranger)
```

```
##
## Attaching package: 'ranger'

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

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

```r
set.seed(12345)
modrf <- ranger(classe~., training3)
predrf <- predict(modrf, testing3)[[1]]
Comprf <- confusionMatrix(predrf, testing3$classe)
Comprf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    6    0
##          D    0    0    0  958    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9938   0.9991
## Specificity            0.9998   0.9996   0.9988   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9942   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1628   0.1837
## Detection Prevalence   0.2846   0.1937   0.1750   0.1630   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9984   0.9968   0.9995
```

```
plot(Comprf$table, col = Comprf$table)
```



As we can see, the accuracy of the random forest model we made is 0.9983.

Boosting

Finally, we'll use the Boosting method to do the prediction.

```
set.seed(12345)
modboost  <- gbm(classe~., data = training3)
```

```
## Distribution not specified, assuming multinomial ...
```

```
## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

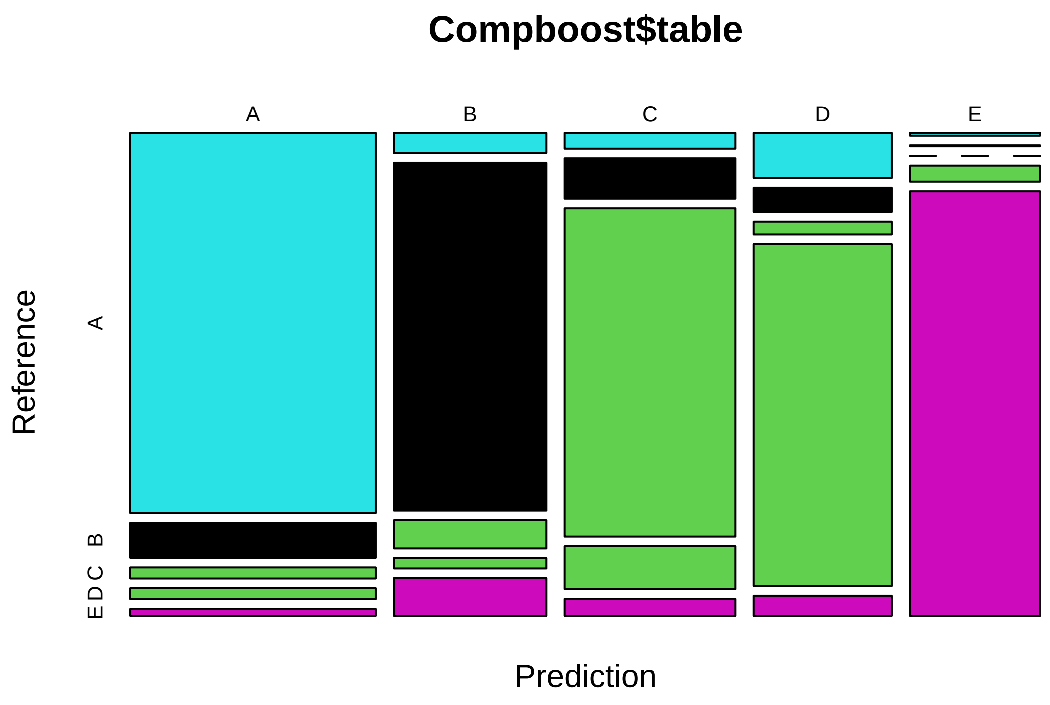
```
predboost0 <- predict.gbm(modboost, testing3,type = "response")
```

```
## Using 100 trees...
```

```
predboost <- factor(colnames(predboost0)[apply(predboost0, 1, which.max)])
Compboost <- confusionMatrix(factor(predboost), testing3$classe)
Compboost
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1477  136   43   43   27
##          B   49  839   68   25   91
##          C   43  109  887  116   46
##          D   99   53   28  747   44
##          E    6    2    0   33  874
##
## Overall Statistics
##
##                Accuracy : 0.8197
##                  95% CI : (0.8096, 0.8295)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7718
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8823   0.7366   0.8645   0.7749   0.8078
## Specificity            0.9409   0.9509   0.9354   0.9545   0.9915
## Pos Pred Value         0.8557   0.7826   0.7386   0.7693   0.9552
## Neg Pred Value         0.9526   0.9377   0.9703   0.9558   0.9581
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2510   0.1426   0.1507   0.1269   0.1485
## Detection Prevalence   0.2933   0.1822   0.2041   0.1650   0.1555
## Balanced Accuracy      0.9116   0.8438   0.9000   0.8647   0.8996
```

```
plot(Compboost$table, col = Compboost$table)
```



As we can see the boosting method has an accuracy of 0.8233.

Model selection

Appearently, the Random Forest model showed the best accuracy as 0.9983. Therefore, we'll use the Random Forest model to predict the testpml dataset.

Applying the Random Forest model on the testpml dataset

```
predrmpml <- predict(modrf, testpml3)[[1]]
predrmpml
```

```
##  [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
```

Therefore, the predicted value for the 20 observations in the test set would be B A B A A E D B A A B C B A E E A B B B.