Prediction of exercise class

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit 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 be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five ways are exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Only Class A corresponds to correct performance. The goal of this project is to predict the manner in which they did the exercise, i.e., Class A to E.

Exploratory data analysis

The data is separated into training and test sets. The test set will not be used in any exploratory analysis however in order to have a better understanding of the data, we look at some of the properties of the train set.

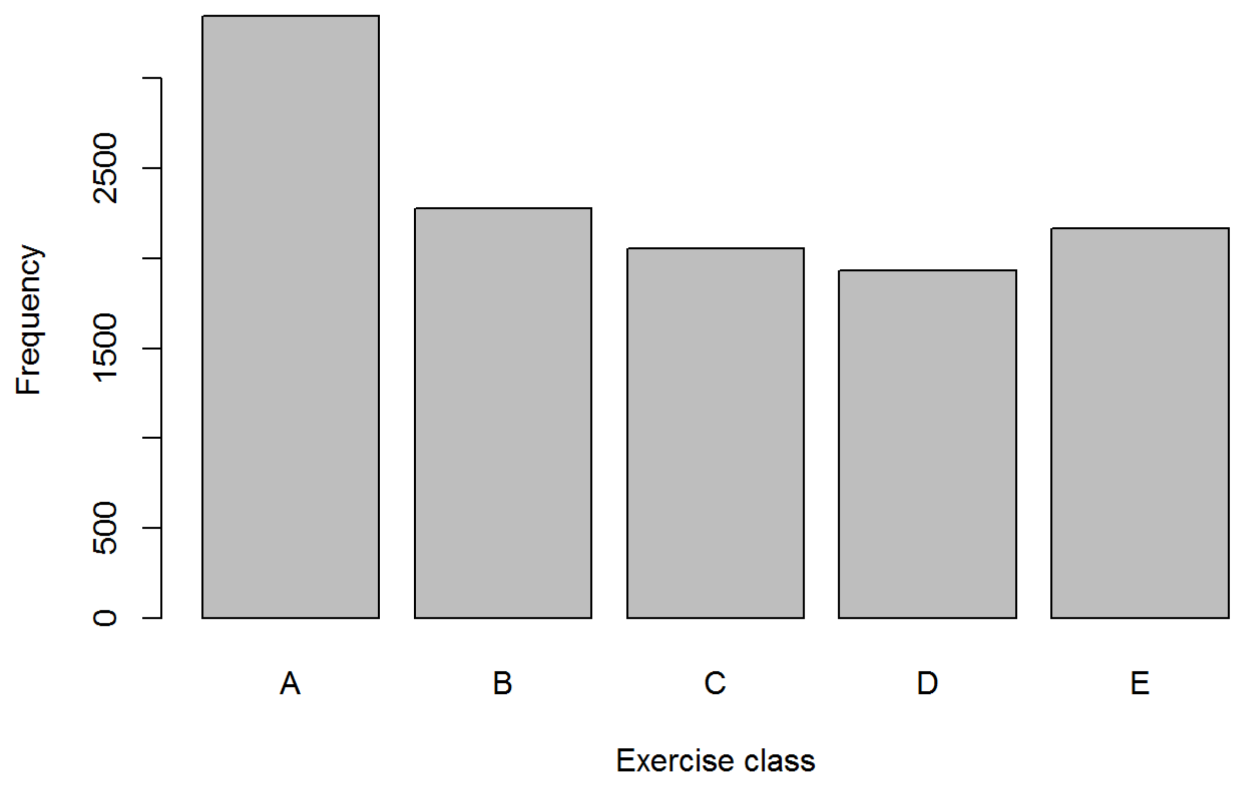
```
library(caret)
library(randomForest)
library(gbm)
library(rpart)
library(rattle)
library(rpart.plot)
trainset <- read.csv("pml-training.csv")
testset  <- read.csv("pml-testing.csv")
dim(trainset)
table(trainset$classe)
```

```
## [1] 19622   160
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

The train set consists of 19622 observations of 160 variables, however the first 7 variables are not going to be used and therefore are removed from both datasets.
Regressors with near zero variance have also been removed from the training set.
Another needed cleanup is to remove regressors that have a lot of NAs. This reduces the number of regressors to 53.

The attribute that is going to be modelled and predicted is `classe` which has 5 factor levels with the distribution shown below.

```
plot(mytrainset$classe, xlab="Exercise class", ylab="Frequency")
```



Data cleaning

```
trainset <- trainset[,-c(1:7)]
testset  <- testset[,-c(1:7)]
nzv <- nearZeroVar(trainset, saveMetrics= TRUE)
mytrainset <- trainset[,!nzv$nzv]
testset <- testset[,!nzv$nzv]
ind <- as.data.frame(apply(mytrainset, 2, function(x) sum(is.na(x))/length(x)<0.7))
mytrainset <- mytrainset[,ind[[1]]]
testset <- testset[,ind[[1]]]
```

Subsetting training data

```
set.seed(12345)
intrain <- createDataPartition(mytrainset$classe, p=0.6, list=FALSE)
mytestset <- mytrainset[-intrain,]
mytrainset <- mytrainset[intrain,]
```

Model building

Random Forest and Decision tree algorithms are examined in this section. All regressors are going to be used.

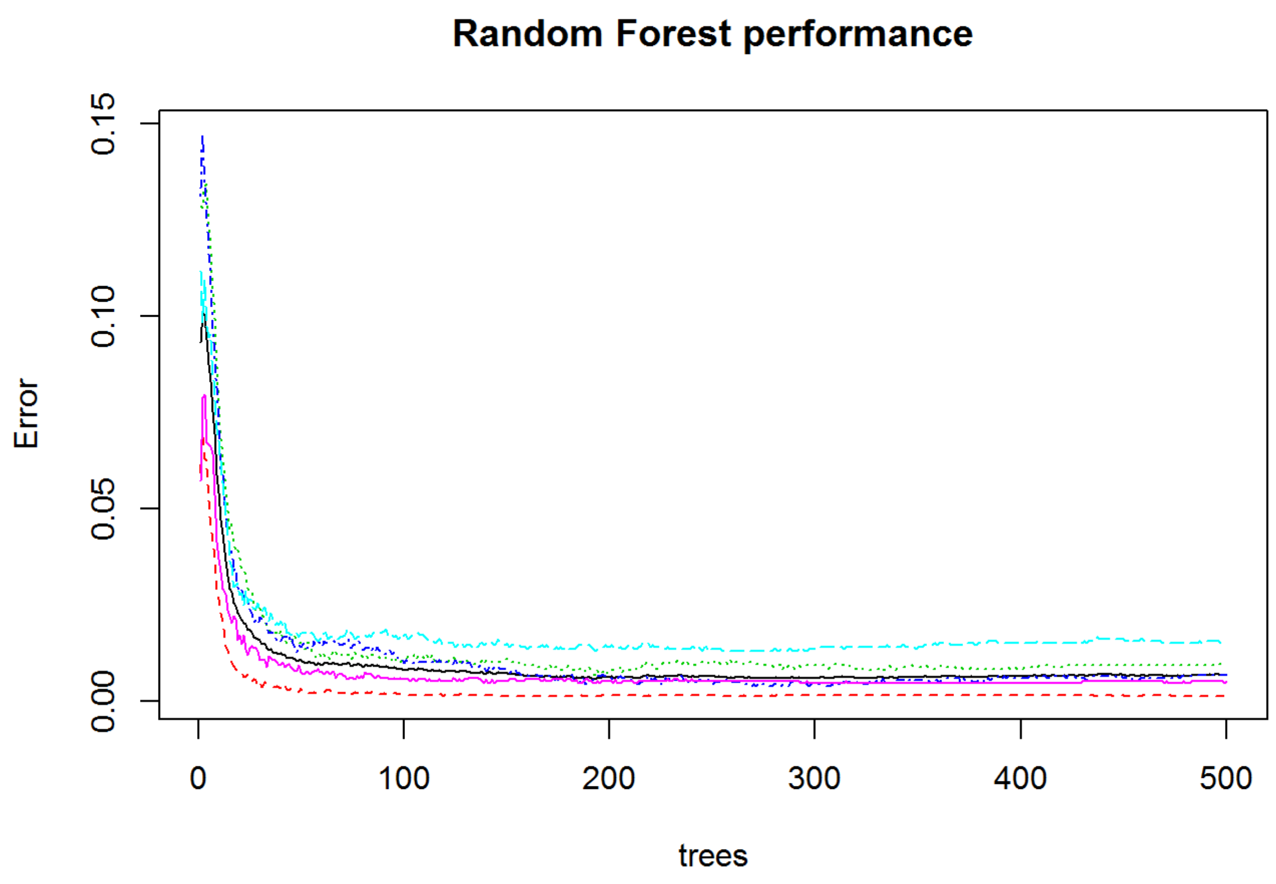
cross validation

In order to have an estimate of out of sample error, without applying the model to the test set, cross validation using a subset of the training set is applied. There are several methods available for cross validating the model, such as leave out one sample, bootstrapping, k folding, etc. In this section the method of data splitting is used.

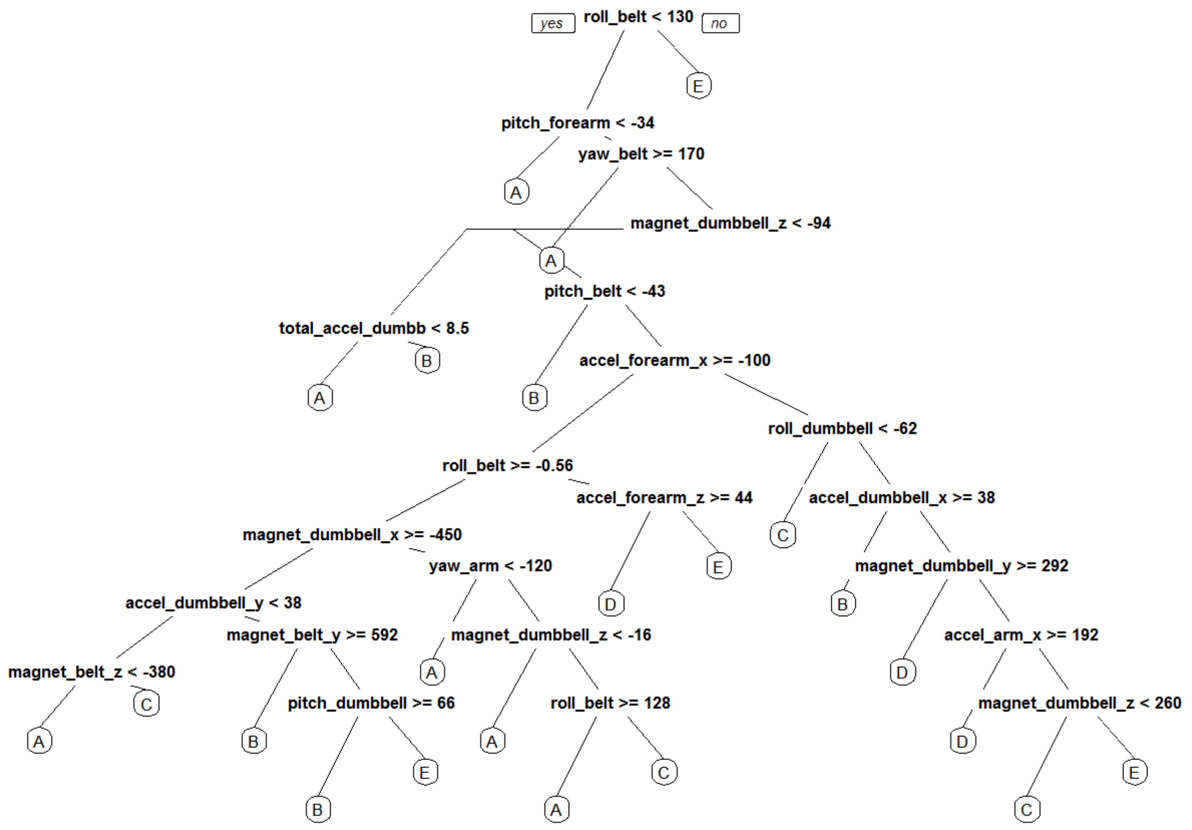
model training

```
set.seed(12345)
fitrf <- randomForest(classe~., data=mytrainset)
fitct <- rpart(classe~., data=mytrainset)
```

```
plot(fitrf, main = "Random Forest performance") #plotting the performance of random forest model
```



```
prp(fitct) #plotting the decision tree structure
```



estimation of out of sample accuracy

```
prdrf <- predict(fitrf, mytestset)
prdct <- predict(fitct, mytestset)

confmrf <- confusionMatrix(prdrf, mytestset$classe)
confmct <- confusionMatrix(prdct, mytestset$classe)

data.frame(RandomForest=confmrf$overal, DecisionTree=confmct$overal)
```

```
##                RandomForest DecisionTree
## Accuracy          0.9945195 3.675758e-01
## Kappa             0.9930674 1.266022e-01
## AccuracyLower     0.9926248 3.568945e-01
## AccuracyUpper     0.9960310 3.783556e-01
## AccuracyNull      0.2844762 2.844762e-01
## AccuracyPValue    0.0000000 3.295419e-57
## McnemarPValue           NaN          NaN
```

The table above shows that the random forest method is expected to be more accurate that the decision tree methos.
The accuracy of RF method is 0.9945 with the 95% CI of 0.9926 to 0.9964.

Predicting the class for the test data

```
prd <- predict(fitrf, testset[,-53])
print(prd)
```

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