Practile machine learning project

Saurabh Yadav

Executive summary

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 and we have to predict that way.

Required library

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.1
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.5.1
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.5.1
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.1
```

```
## 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
```

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

```
library(rpart)
```

Loading the data

```
trainingdata<-read.csv("pml-training.csv")
testingdata<-read.csv("pml-testing (1).csv")
```

Cleaning the data first

removing the column having the NAs more than 90% of the colomn data

```
colindtraining<-which(colSums(is.na(trainingdata)|trainingdata=="")>0.9*dim(trainingdata)[1])
colindtesting<-which(colSums(is.na(testingdata)|testingdata=="")>0.9*dim(testingdata)[1])
cleantraining<-trainingdata[,-colindtraining]
cleanedtraining<-cleantraining[,-c(1:7)]

#cleaned testing data set

cleantesting<-testingdata[,-colindtesting]
newtesting<-cleantesting[,-1]
```

Dividing the cleanedtraining data set in to training set and test set

```
intrain<-createDataPartition(y=cleanedtraining$classe,p=0.7,list = FALSE)
training<-cleanedtraining[intrain,]
testing<-cleanedtraining[-intrain,]
```

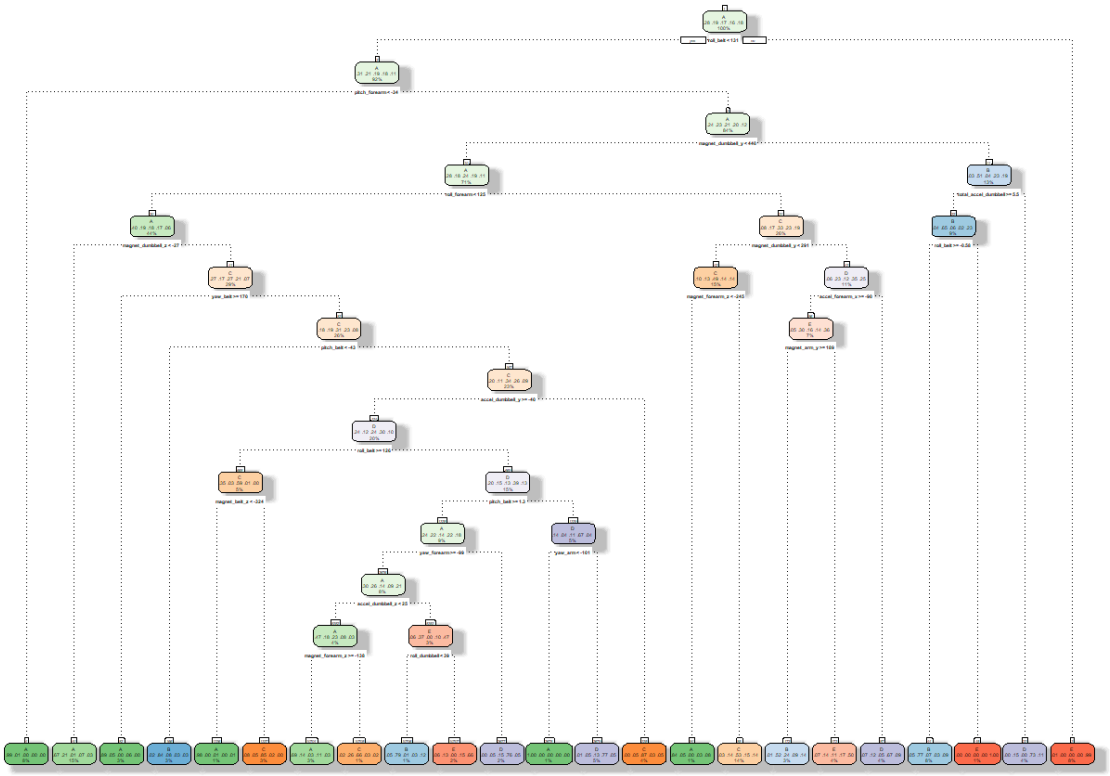
Fitting the model

1. Fitting the tree model

```
modfittree<-rpart(classe~.,data=training,method = "class")
```

2. Plotting the tree

```
fancyRpartPlot(modfittree)
```



Rattle 2018-Sep-23 17:39:53 SAURABH YADAV

####3.

predicting with tree

```
pred<-predict(modfittree,testing,type = "class")
```

4.Checking the accuracy

```
confusionMatrix(pred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1533  230   18  120   43
##          B   37  607   71   36   62
##          C   57  159  828  146  140
##          D   15   98   85  599   41
##          E   32   45   24   63  796
##
## Overall Statistics
##
##                Accuracy : 0.7414
##                  95% CI : (0.73, 0.7525)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6712
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9158   0.5329   0.8070   0.6214   0.7357
## Specificity            0.9024   0.9566   0.8967   0.9514   0.9659
## Pos Pred Value         0.7886   0.7466   0.6226   0.7148   0.8292
## Neg Pred Value         0.9642   0.8951   0.9565   0.9277   0.9419
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2605   0.1031   0.1407   0.1018   0.1353
## Detection Prevalence   0.3303   0.1381   0.2260   0.1424   0.1631
## Balanced Accuracy      0.9091   0.7448   0.8519   0.7864   0.8508
```

Conclusion: The accuracy is only 73% which is not that good. So we need another model to predict.

Creating model with random forest

```
modfitrf<-randomForest(classe~.,data=training)
```

1.predicting with random forest

```
p<-predict(modfitrf,testing)
```

2. checking the accuracy

```
confusionMatrix(p,testing$classe)
```

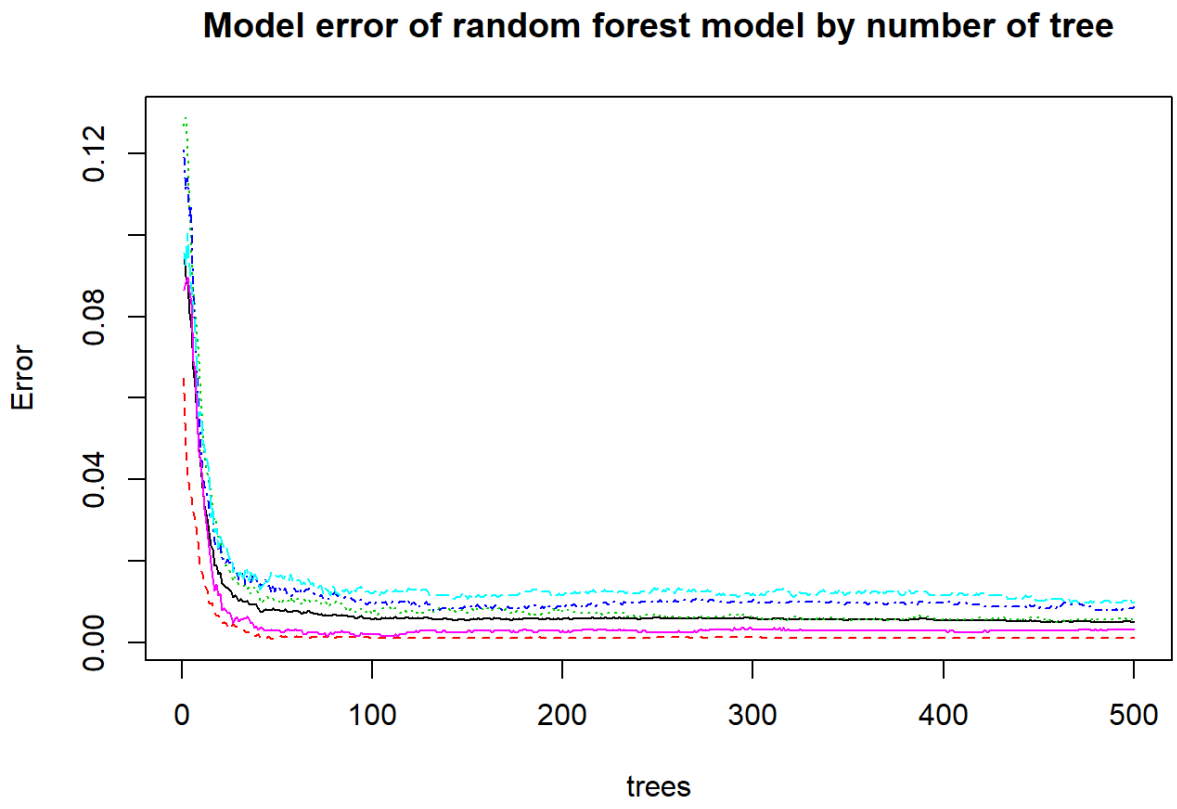
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    0 1131    1    0    0
##          C    1    1 1022    9    4
##          D    0    0    3  952    0
##          E    1    0    0    3 1078
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9961   0.9876   0.9963
## Specificity            0.9983   0.9998   0.9969   0.9994   0.9992
## Pos Pred Value         0.9958   0.9991   0.9855   0.9969   0.9963
## Neg Pred Value         0.9995   0.9983   0.9992   0.9976   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1737   0.1618   0.1832
```

```
## Detection Prevalence   0.2853   0.1924   0.1762   0.1623   0.1839
## Balanced Accuracy      0.9986   0.9964   0.9965   0.9935   0.9977
```

Conclusion : The random forest model has the best accuracy which is 99.92%. So this would be the model to predit.

Model error of random forest model

```
plot(modfitrf,main="Model error of random forest model by number of tree")
```



Applying the model for final prediction on the test data set

```
predict(modfitrf,newtesting)
```

```
##  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
```