Practical Machine Learning Project

Sarim

20/06/2020

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. In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

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.

We will be using decision trees and random forest methods for predictions

Data Processing

Libraries Used

```
library(caret)
```

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

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

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

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

Loading Data

```
#Before doing any operations we set the seed to maintain reproducibility
set.seed(12345)
training<-read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!", ""))
testing<-read.csv("pml-testing.csv",na.strings=c("NA","#DIV/0!", ""))
```

Cleaning Data

The datasets loaded have a few columns which have no results and the first 7 columns have no information which will help us in the predictions. So we remove these columns for both the datasets. Also the predicted variable *classe* is a factor but present as a character and so it is converted into a factor variable.

```
training<-training[,colSums(is.na(training))==0]
testing<-testing[,colSums(is.na(testing))==0]
training<-training[,-c(1:7)]
testing<-testing[,-c(1:7)]

# Converting classe to factor type
training$classe<-as.factor(training$classe)
```

Partitioning of data

The testing set is kept aside for final test just once and the training set is divided into two subsets - for training and cross validation. The two datasets are created using *createDataPartition* with a prob distribution of 60% and 40% in favour of training.

```
inTrain<-createDataPartition(training$classe,p=0.6,list=FALSE)
newTraining<-training[inTrain,]
newTesting<-training[-inTrain,]
```

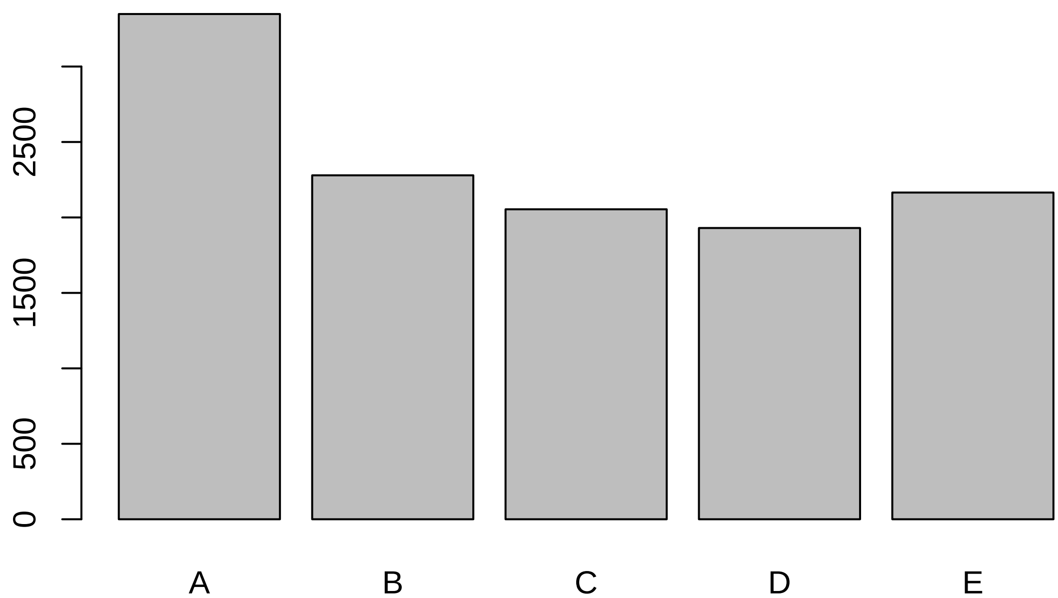
Prediction

Before diving into the prediction we first look the variable we are dealing with and do some simple analysis

```r
summary(newTraining$classe)
```

```
##    A    B    C    D    E 
## 3348 2279 2054 1930 2165
```

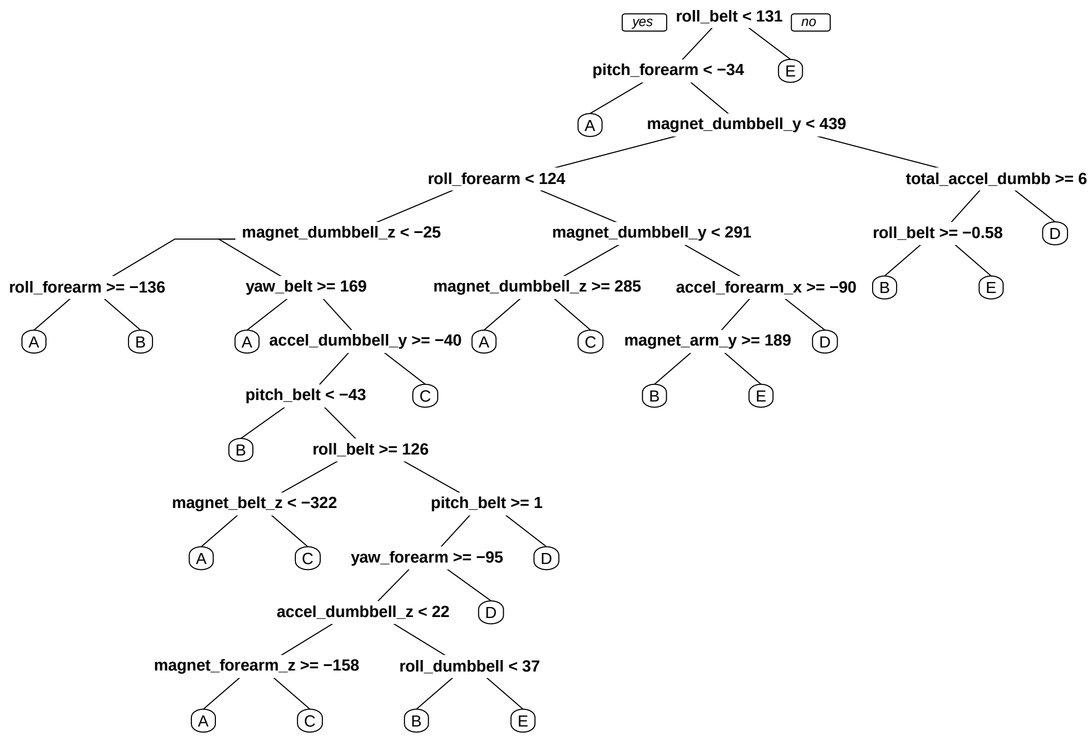
```r
plot(newTraining$classe)
```



From the plot we can see that *classe* is factor with 5 levels *A,B,C,D and E*. *A* level is significantly more while all others are close to each other.

Decision Trees using rpart

```r
fit1<-rpart(classe ~ ., data=newTraining, method="class")
prp(fit1)
```




```r
pred1<-predict(fit1,newTesting,type = "class")
confusionMatrix(pred1,newTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1995  246   49   83   51
##          B   75  890  111  119  115
##          C   44  198 1094  153  143
##          D   74  112   79  840   94
##          E   44   72   35   91 1039
##
## Overall Statistics
##
##                Accuracy : 0.7466
##                  95% CI : (0.7368, 0.7562)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6787
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8938   0.5863   0.7997   0.6532   0.7205
## Specificity            0.9236   0.9336   0.9169   0.9453   0.9622
## Pos Pred Value         0.8230   0.6794   0.6703   0.7006   0.8111
## Neg Pred Value         0.9563   0.9039   0.9559   0.9329   0.9386
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2543   0.1134   0.1394   0.1071   0.1324
## Detection Prevalence   0.3089   0.1670   0.2080   0.1528   0.1633
## Balanced Accuracy      0.9087   0.7600   0.8583   0.7992   0.8414
```

As can be seen by the result of the confusion matrix this is not a very good predictor and gives an accuracy of about 75%.This model does not have good sensitivity but has decent specificity. So we look for a better method if possible.

Random Forest

```
fit2<-randomForest(classe ~.,newTraining)
pred2<-predict(fit2,newTesting,type = "class")
confusionMatrix(pred2,newTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    3    0    0    0
##          B    3 1515    6    0    0
##          C    0    0 1357    6    2
##          D    0    0    5 1280    5
##          E    0    0    0    0 1435
##
## Overall Statistics
##
##                Accuracy : 0.9962
##                  95% CI : (0.9945, 0.9974)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9952
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9980   0.9920   0.9953   0.9951
## Specificity            0.9995   0.9986   0.9988   0.9985   1.0000
## Pos Pred Value         0.9987   0.9941   0.9941   0.9922   1.0000
## Neg Pred Value         0.9995   0.9995   0.9983   0.9991   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1931   0.1730   0.1631   0.1829
```

```
## Detection Prevalence   0.2845   0.1942   0.1740   0.1644   0.1829
## Balanced Accuracy      0.9991   0.9983   0.9954   0.9969   0.9976
```

The random forest method provides great result with accuracy over 99%. This method is great for our data and no further improvement looks likely and so this method can be used for any further predictions related to this type of data.

Final Test predictions

```
pred_test<-predict(fit2,testing,type = "class")
```