Practical Machine Learning

Mrugank Akarte

25 September 2016

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. 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).

The data for this project come from this source.

Loading and cleaning Data

```
training<-read.csv("pml-training.csv",na.strings = c("NA","","#DIV/0!"))
testing<-read.csv("pml-testing.csv", na.strings = c("NA","#DIV/0!"))
```

Training data set contains 19622 observations of 160 variables. Testing data set contains 20 observations of 160 variables.

Removing columns which contains more than 60% NA values.
Also removing the first 7 columns which might interfer in algorithms.

```
training<- training[ , (colSums(is.na(training))/nrow(training)) < 0.6]
testing<-testing[, (colSums(is.na(testing))/nrow(testing)) < 0.6]
training<-training[,-c(1:7)]
testing<-testing[,-c(1:7)]
```

Partitioning data into training and testing data set.

```
intrain<-createDataPartition(training$classe, p= 0.6, list=F)
traindata<-training[intrain,]
testdata<-training[-intrain,]
```

The data is now clean and ready for modelling.

Modelling the Data

Decision Tree

```
model2 <- train(classe ~ ., data = traindata, method = "rpart")
```

Estimating performance on testdata.

```
predict2 <- predict(model2, testdata)
confusionMatrix(testdata$classe, predict2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1326  248  484  165    9
##          B  248  851  342   77    0
##          C   37   58 1077  196    0
##          D   81  178  689  338    0
##          E   19  328  346   90  659
##
## Overall Statistics
##
##                Accuracy : 0.5418
##                  95% CI : (0.5307, 0.5529)
##     No Information Rate : 0.3745
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4258
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7750   0.5117   0.3666  0.39030  0.98653
## Specificity            0.8523   0.8921   0.9407  0.86418  0.89092
## Pos Pred Value         0.5941   0.5606   0.7873  0.26283  0.45700
## Neg Pred Value         0.9314   0.8717   0.7127  0.91951  0.99859
## Prevalence             0.2181   0.2120   0.3745  0.11037  0.08514
## Detection Rate         0.1690   0.1085   0.1373  0.04308  0.08399
## Detection Prevalence   0.2845   0.1935   0.1744  0.16391  0.18379
## Balanced Accuracy      0.8137   0.7019   0.6536  0.62724  0.93872
```

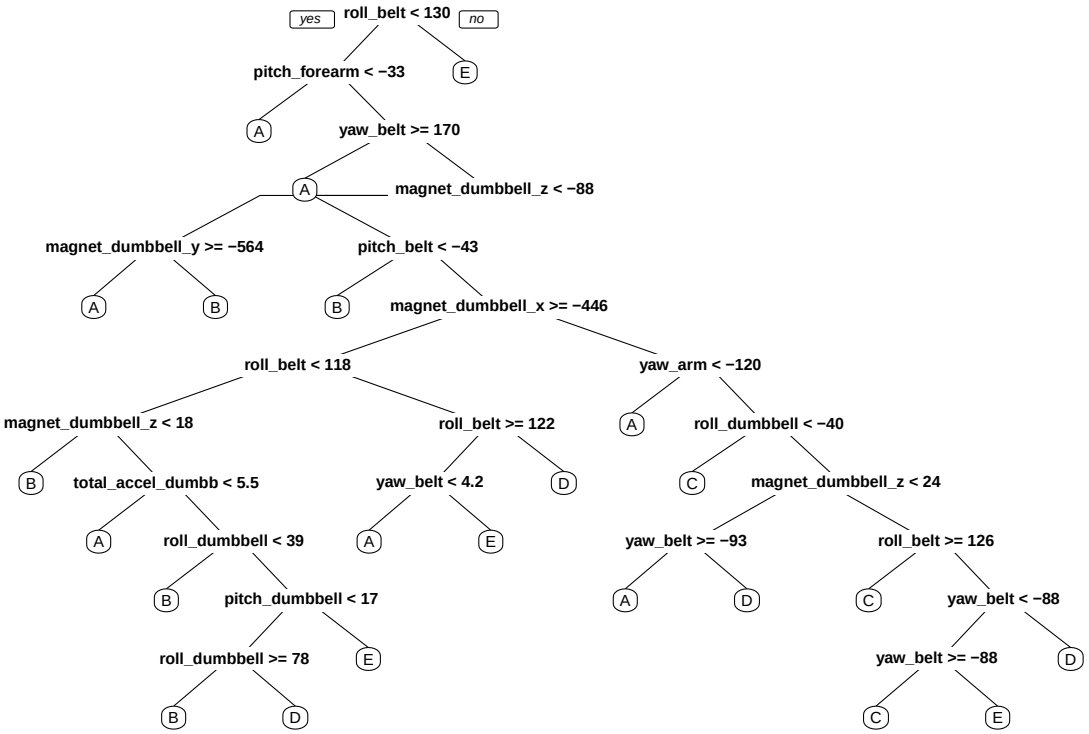
```
oose<- 1 -  confusionMatrix(testdata$classe, predict2)$overall[1]
oose
```

```
##  Accuracy
## 0.4581953
```

Accuracy of decision tree is around 55% hence out of sample error is around 45%. Thus predictions using this method will not yield good results.

Pictorical representation of decision tree.

```
tree <- rpart(classe ~ ., data = traindata, method = "class")
prp(tree)
```



Random Forest

For random forests algorithm **5-Fold cross-validation** was used.

```
controltr <- trainControl(method = "cv", 5)
model1 <- train(classe ~ ., data = traindata, method = "rf", trControl = controltr, ntree = 250)
```

Estimating performance on testdata.

```
predict1 <- predict(model1, testdata)
confusionMatrix(testdata$classe, predict1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2223    9    0    0    0
##          B   15 1497    6    0    0
##          C    0   14 1348    6    0
##          D    0    1   29 1254    2
```

```
##          E    0    0    7    5 1430
##
## Overall Statistics
##
##                Accuracy : 0.988
##                  95% CI : (0.9854, 0.9903)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9848
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9933   0.9842   0.9698   0.9913   0.9986
## Specificity            0.9984   0.9967   0.9969   0.9951   0.9981
## Pos Pred Value         0.9960   0.9862   0.9854   0.9751   0.9917
## Neg Pred Value         0.9973   0.9962   0.9935   0.9983   0.9997
## Prevalence             0.2852   0.1939   0.1772   0.1612   0.1825
## Detection Rate         0.2833   0.1908   0.1718   0.1598   0.1823
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9958   0.9905   0.9833   0.9932   0.9984
```

```
oose<- 1 - confusionMatrix(testdata$classe, predict1)$overall[1]
oose
```

```
##   Accuracy
## 0.01198063
```

Accuracy of this model is very high comapred to decision tree. Out of sample error is close to 0.5%. Thus using Random forests instead of decision tree will yield much better results.

Predicting for Testing data set

Predicting the *classe* from Testing data set.

```
predict_test<-predict(model1, testing)
predict_test
```

```
##  [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
```