Machine Learning Prediction project

Sam G

Feb 22, 2015

Introduction:

As part of **Coursera Machine Learning Course** this Prediction Assignment uses data from a personal activity monitoring device(Fitbit, Nike Fuelband, or Jawbone Up), to predict the manner in which participants did the exercise.

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).

Synopsis:

The goal of this project is to predict the manner in which the participants did the exercise. This is the "classe" variable in the training set. And may use any of the other variables to predict with.

Data:

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"

Setting up the environment and downloading file

```r
setwd ("~/scientist/maclearn")

download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', method= 'curl', 'p

download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv', method= 'curl', 'pr

library(ggplot2)
library(lattice)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Reading the Training and Testing Data:

```r
traindata <- read.csv('~/scientist/maclearn/pml-training.csv')
testdata <- read.csv('~/scientist/maclearn/pml-testing.csv')
dim(traindata) # Getting the dimension
```

```
## [1] 19622   160
```

```r
dim(testdata) # Getting the dimension
```

```
## [1]  20 160
```

As we can see the training data set has 19622 observations and 160 variables whereas testing data set contains 20 observations and 160 variables. The "classe" variable in the training set is the outcome to predict.

Filtering the data:

Removing the missing values - NA and unnecessary variables,info etc.

```r
traindata <- traindata[, colSums(is.na(traindata)) == 0]
testdata <- testdata[, colSums(is.na(testdata)) == 0]

sum(complete.cases(traindata))
```

```
## [1] 19622
```

```r
sum(complete.cases(testdata))
```

```
## [1] 20
```

```r
classe <- traindata$classe
trainfilter <- grepl("^X|timestamp|window", names(traindata))
traindata <- traindata[, !trainfilter]
traingood <- traindata[, sapply(traindata, is.numeric)]
traingood$classe <- classe
testfilter <- grepl("^X|timestamp|window", names(testdata))
testdata <- testdata[, !testfilter]
testgood <- testdata[, sapply(testdata, is.numeric)]
```

```
dim(testgood)
```

```
## [1] 20 53
```

```
dim(traingood)
```

```
## [1] 19622    53
```

We can see the filtered data has 19622 observations and 53 vairables whereas testing data set has 20 observations and 53 vaiables. The `classe` variable is part of this filtered data.

Segregating the Data:

At this stage we'll segregate the filtered data into only training data set which will be 80% and a validatition set as 20%. The validatition set will be used for a check on our assumptions and inferences.

```
set.seed(22519)
Justtrain <- createDataPartition(traingood$classe, p=0.80, list=F) #data splitting function of "caret"
Trfinal <- traingood[Justtrain, ]
Tefinal <- traingood[-Justtrain, ]
```

Algorithm for Predictive Data Model:

The algorihm we'll be suing will have to fit our prective analysis. For this we'll be using `Random Forest` algorithm as they are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set.

```
RanForStage <- trainControl(method="cv", 5) #trainControl to control the computational nuances of the t
#5 fold cross validation is used here "http://en.wikipedia.org/wiki/Cross-validation_%28statistics%29"

RanForMod <- train(classe ~ ., data=Trfinal, method="rf", trControl=RanForStage, ntree=250)
RanForMod
```

```
## Random Forest
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 12558, 12560, 12559, 12559, 12560
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9924835  0.9904916  0.001893201  0.002394685
##   27    0.9927384  0.9908140  0.001448814  0.001832282
##   52    0.9864322  0.9828362  0.001434944  0.001815461
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Once we have the Random Forest function performing sampling and processing, we'll estimate the preformance of the model.

```
RanForPred <- predict(RanForMod, Trfinal)
confusionMatrix(Trfinal$classe, RanForPred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Now we test the accuracy of our predition.

```
Acurcy <- postResample(RanForPred, Trfinal$classe)
Acurcy
```

```
## Accuracy    Kappa
##        1        1
```

```
SpleErr <- 1 - as.numeric(confusionMatrix(Trfinal$classe, RanForPred)$overall[1])
SpleErr #Getting Sample error
```

```
## [1] 0
```

We see the estimated accuracy of the model is 100% and the sample error is 0%.

Final Prediction of the Test Data Set:

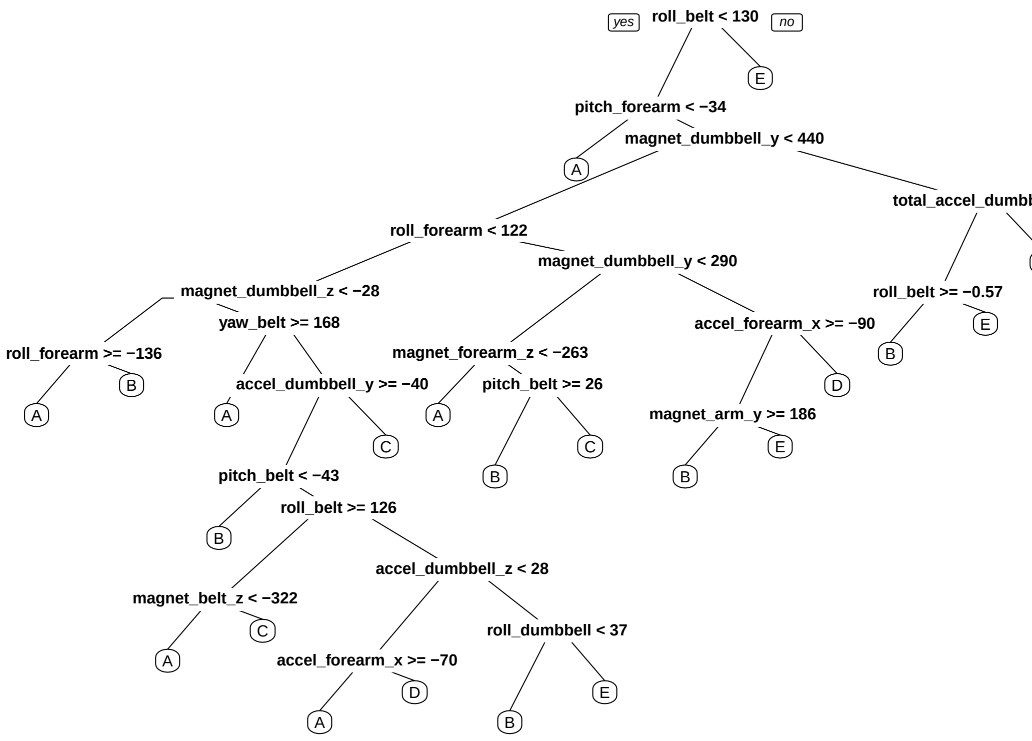
```
TestResult <- predict(RanForMod, testgood[, -length(names(testgood))])
TestResult
```

```
##  [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
```

Appendix:

```


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



Model of the Prediction Tree.