Practical Machine Learning: Project

RK

January 30, 2016

Introduction

The goal of this project is to predict the label quantifying how well a physical activity is performed by a participant using a test data set that consists of accelerometer readings from devices such as Fitbit placed strategically across the participant's body. The training data set given to us consists of accelerometer readings along with the value of the quantifying label.

Download,Read and Clean/Preprocess the datasets

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              "./TrainingData.csv", method = "curl")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              "./TestData.csv", method = "curl")
## A quick peek at the data showed that there are invalid values like #DIV/0! and empty strings.
## We interpret them as NAs while reading the data.
dfTrain <- read.table("./TrainingData.csv", sep=",", header=TRUE, na.strings=c("NA","#DIV/0!", ""))
dfTest <- read.table("./TestData.csv", sep=",", header=TRUE, na.strings=c("NA","#DIV/0!", ""))
```

The first seven columns don't specify any accelerometer data and so they won't contrubute towards the prediction . Removing them here ->

```
dfTrain <- dfTrain[,-(1:7)]
## Repeat for the test data set
dfTest <- dfTest[,-(1:7)]
```

Next , we remove columns whose values are almost always NA .

```
flag <- sapply(dfTrain, function(x) mean(is.na(x))) > 0.80
dfTrain <- dfTrain[,flag==FALSE]
## Repeat for the test data set. Remove the same variables in the
## test data set that we removed in the train data set.We are using
## just dfTrain as an argument in sapply above and using the same flag
## to filter the test data set.
dfTest <- dfTest[,flag==FALSE]
```

Also , let us set the seed for reproducability before moving ahead

```
set.seed(2016)
```

Partition the training data set

Next , we partition the training data set into two parts that would later help us find the accuracy and the out-of-sample error using a confusion matrix.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

```
trainingDataMatrix <- createDataPartition(dfTrain$classe, p=0.75, list=FALSE)
dfTrainA <- dfTrain[trainingDataMatrix, ]
dfTrainB <- dfTrain[-trainingDataMatrix, ]
```

Random Forest Model

Next , we build a random forest model using dfTrainA and evaluate the model.

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rfm <- train(classe ~ ., data=dfTrainA,
             method="rf",
             trControl=trainControl(method="cv", number=3, verboseIter=FALSE))
rfm$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.67%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4176    6    2    0    1 0.002150538
## B   19 2824    5    0    0 0.008426966
## C    0    9 2550    8    0 0.006622517
## D    0    1   29 2381    1 0.012852405
## E    0    2    5   10 2689 0.006282336
```

Now , we evaluate the above model by predicting “classe” using dfTrainB and use the confusion matrix to get an estimate of the accuracy.

```
confusionMatrix(dfTrainB$classe, predict(rfm, newdata=dfTrainB))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    6  942    1    0    0
##          C    0    3  845    7    0
##          D    0    0    6  798    0
##          E    2    0    2    4  893
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9908, 0.9955)
##     No Information Rate : 0.2859
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9917
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9958   0.9895   0.9864   1.0000
## Specificity            0.9997   0.9982   0.9975   0.9985   0.9980
## Pos Pred Value         0.9993   0.9926   0.9883   0.9925   0.9911
## Neg Pred Value         0.9977   0.9990   0.9978   0.9973   1.0000
## Prevalence             0.2859   0.1929   0.1741   0.1650   0.1821
## Detection Rate         0.2843   0.1921   0.1723   0.1627   0.1821
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9970   0.9970   0.9935   0.9925   0.9990
```

From the confusion matrix results above , the random forest model yields us an accuracy of 0.9935 and so we continue with using this model for our prediction .

```
##Train the entire dfTrain set using our selected model.
##Also , because the random forest model is computationally
##intensive , we are setting 'cache=TRUE' for all computationally
##intensive blocks in R markdown.
predictionModel <- train(classe ~ .,
                         data=dfTrain,
                         method="rf",
                         trControl=trainControl(method="cv", number=3, verboseIter=FALSE))
```

Test Set prediction

Finally , the selected model is applied to the test data to predict the labels .

```
testDataPredictions <- predict(predictionModel, newdata=dfTest)
testDataPredictions
```

```
##  [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
```