Practical Machine Learning Course Project

twgoh

Sunday, March 22, 2015

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).

Data

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. We load the two datasets:

```
train <- read.csv("C:/Users/User/Documents/Coursera/08 - Practical Machine Learning/pml-training.csv", 
test <- read.csv("C:/Users/User/Documents/Coursera/08 - Practical Machine Learning/pml-testing.csv", na
```

The training data has 19622 observations and 160 features, and the distribution of the five measured stances A,B,C,D,E is:

```
dim(train)
```

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

```
table(train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Preprocessing

```
library(caret)
```

Cleaning the data

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

```
#remove columns that are not relevant for classification
train <- train[,7:ncol(train)]

#remove columns with over a 90% of not a number
nasPerColumn<- apply(train,2,function(x) {sum(is.na(x))})
train <- train[,which(nasPerColumn <  nrow(train)*0.9)]

#remove near zero variance predictors
nearZeroColumns <- nearZeroVar(train, saveMetrics = TRUE)
train <- train[, nearZeroColumns$nzv==FALSE]

#class into factor
train$classe <- factor(train$classe)
```

Partitioning the training set We separate our training data into a training set and a validation set so that we can validate our model.

```
set.seed(123)
trainset <- createDataPartition(train$classe, p = 0.8, list = FALSE)
Training <- train[trainset, ]
Validation <- train[-trainset, ]
```

Create Machine Learning Models

Three models are generated: random forest ("rf"), boosted trees ("gbm") and linear discriminant analysis ("lda") model. We use the random forest model.

```
library(parallel)
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 3.1.3

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.1.3

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 3.1.3
```

```
registerDoParallel(makeCluster(detectCores()))

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3

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

```
rfModel <- randomForest(classe ~ ., data = Training, importance = TRUE, ntrees = 10)
#model_gbm <-train(classe ~ ., method = 'gbm', data = Training)
#model_lda <-train(classe ~ ., method = 'lda', data = Training)
```

Model Validation

Let us now test our model performance on the training set itself and the cross validation set.

```
ptraining <- predict(rfModel, Training)
print(confusionMatrix(ptraining, Training$classe))
```

Training set accuracy

```
## 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
```

Obviously our model performs excellent against the training set, but we need to cross validate the performance against the held out set and see if we have avoided overfitting.

Validation set accuracy (Out-of-Sample) Let us now see how our model performs on the cross validation set that we held out from training.

```
pvalidation <- predict(rfModel, Validation)
print(confusionMatrix(pvalidation, Validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  757    1    0    0
##          C    0    0  683    2    0
##          D    0    0    0  641    1
##          E    0    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9967, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9985   0.9969   0.9986
## Specificity            0.9993   0.9997   0.9994   0.9997   1.0000
## Pos Pred Value         0.9982   0.9987   0.9971   0.9984   1.0000
## Neg Pred Value         1.0000   0.9994   0.9997   0.9994   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1741   0.1634   0.1835
## Detection Prevalence   0.2850   0.1932   0.1746   0.1637   0.1835
## Balanced Accuracy      0.9996   0.9985   0.9990   0.9983   0.9993
```

The cross validation accuracy is 99.85%, so our model performs rather good.

Test set prediction

The prediction of our algorithm for the test set is:

```
ptest <- predict(rfModel, test)
ptest
```

```
##  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
```

We then save the output to files according to instructions and post it to the submission page.

```
answers <- as.vector(ptest)

pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(answers)
```