Prediction Assignment Writeup

Yao Wang

Friday, November 21, 2014

Background

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).

Getting data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

```
trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
download.file(trainurl, destfile = "~/machine//project/pml-training.csv",
method = "curl", setInternet2(TRUE))
pmltraining <- read.csv("~/machine//project/pml-training.csv")
```

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
download.file(testurl, destfile = "~/machine//project/testing.csv", method =
"curl", setInternet2(TRUE))
pmltesting <- read.csv("~/machine//project/testing.csv")

dim(pmltraining); dim(pmltesting)

## [1] 19622   160

## [1]  20 160
```

Checking these datasets, we found there are a lot of missing data. To get a tidy training data, we need to do some cleaning and preprocessing.

Cleaning and Preprocessing the training data

```
library(caret)

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

library(randomForest)

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

We built a function "cleanna" to remove the variables with "NA"s in them:

```
cleanna <- function(dataset){
        n <- ncol(dataset)
        numcol <- numeric()
        for(i in 1:n){if (sum(is.na(dataset[,i]))==0){numcol <- c(numcol, 
i)}}
        dataset[,numcol]
}
```

First we will apply "cleanna" function to training data.

```
nonatrain <- cleanna(pmltraining)
```

Then we will diagnose the near zero variance predictors and remove them.

```
nzvtrain <- nearZeroVar(nonatrain, saveMetrics=T)
traindata <- nonatrain[,nzvtrain$nzv=="FALSE"]
```

The index "X", "user_name", and "timestamp" variables are useless to prediction, so we will remove them.

```
traindata <- traindata[, -c(1:5)]
dim(traindata)

## [1] 19622    54
```

Now the training data has been reduced to 54 columns.

Modeling

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

First, we will subset the training data to two sets. One will be used to built a model, the other will be used to evaluate the model.

```
set.seed(123)
intrain <- createDataPartition(y=traindata$classe, p=0.75, list = FALSE)
trainsub <- traindata[intrain,]; testsub <- traindata[-intrain,]
```

We will use "Random Forests" to train the data. "Random Forests" is one of the two top performing algorithms along with boosting in prediction contest. Although it is difficult to interpret, it is often very accurate.

```
fit <- randomForest(classe ~ ., data = trainsub, importance = F)
fit

##
## Call:
##  randomForest(formula = classe ~ ., data = trainsub, importance = F)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.29%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4184    0    0    0    1   0.0002389
## B    5 2841    2    0    0   0.0024579
## C    0   11 2555    1    0   0.0046747
## D    0    0   15 2397    0   0.0062189
## E    0    0    0    7 2699   0.0025868

varimportance <- varImp(fit)
order(varimportance, decreasing = TRUE)

##  [1]  1  2  4 42 40  3 39 41 38 28 36 11 14 13 37 48  8 15 31 25 30 53 12
## [24] 50 22 35 26 33 17  5 51 52 29 27 16 43 23  9 10 49 24 20 32 19  7 46
## [47] 18  6 44 47 34 45 21
```

Model evaluation

Confusion matrix can be used to evaluate our model.

```
pred <- predict(fit, testsub)
confusionMatrix(testsub$classe, pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    1  948    0    0    0
##          C    0    5  850    0    0
##          D    0    0    9  795    0
##          E    0    0    0    1  900
##
## Overall Statistics
```

```
##
##                Accuracy : 0.997
##                  95% CI : (0.995, 0.998)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.995    0.990    0.999    1.000
## Specificity             1.000    1.000    0.999    0.998    1.000
## Pos Pred Value          1.000    0.999    0.994    0.989    0.999
## Neg Pred Value          1.000    0.999    0.998    1.000    1.000
## Prevalence              0.285    0.194    0.175    0.162    0.184
## Detection Rate          0.284    0.193    0.173    0.162    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    0.997    0.994    0.998    1.000
```

The accuracy of our model is 99.7%.

Predicting testing data with our model

We will predict testing data with the model we built.

```
answers <- predict(fit, newdata = pmltesting)
answers
```

```
##  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
```

Finally, to submit our answer, we will apply this function:

```
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)
  }
}
```

We will create a folder where the files to be written and set that to be the working directory and run:

```
pml_write_files(answers)
```