Machine Learning

Niranjan

Wednesday, Jan 18, 2023

Introduction

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 behaviour, 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 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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Analysis

How the model was build

The following steps were executed to perform prediction with a build model

- Load the test data
- Split the test data again in a test and train set to perform cross-validation
- Explore and clean the data to remove bad predictors
- Build a predictions model on the train set
- Verify the model by predicting the outcomes of the test set created from the train set. If the out of sample error is low we can continue by performing prediction on the real test set.

Loading the data

Loading the training and test data:

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

Cross validation

Cross-validation will be used to split our training set into two sets: a training set and a test set. The model will be fitted with the training set and evaluated with the test set. We will now split up the original training set with a 70/30 ratio:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

```
set.seed(9040)
inTrain <- createDataPartition(y=pmltrain$classe,p=0.7,list=FALSE)
trainData <- pmltrain[inTrain,]
testData <- pmltrain[-inTrain,]
```

Exploring the data

First we try to get an overview of the data:

```
summary(trainData)
```

The results are hidden due to size.

Through this summary we found that the data has columns that have a lot of NA values. We remove all columns with more than 90% of their column filled with NA values as they are not useful predictors.

```
trainData <- trainData[!colSums(is.na(trainData)/nrow(trainData))>.9]
```

Looking at the data we also decided to remove:

- the first column which containing a row index
- the column with the user name
- the columns containing timing when events were logged columns to remove:

```
names(trainData[,grep("X|user_name|*timestamp*",names(trainData))])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"
```

Predictors with very low variance are bad predictors. In a last step we look if we can find such columns using a near zero variance function:

```
nzv <- (nearZeroVar(trainData))
names(trainData)[nzv]
```

```
## [1] "new_window"
```

The data cleaned:

```
trainData <- trainData[,-grep("X|user_name|*timestamp*",names(trainData))]
trainData <- trainData[,-nearZeroVar(trainData)]
names(trainData)
```

```
##  [1] "num_window"          "roll_belt"           "pitch_belt"
##  [4] "yaw_belt"            "total_accel_belt"    "gyros_belt_x"
##  [7] "gyros_belt_y"        "gyros_belt_z"        "accel_belt_x"
## [10] "accel_belt_y"        "accel_belt_z"        "magnet_belt_x"
## [13] "magnet_belt_y"       "magnet_belt_z"       "roll_arm"
## [16] "pitch_arm"           "yaw_arm"             "total_accel_arm"
## [19] "gyros_arm_x"         "gyros_arm_y"         "gyros_arm_z"
## [22] "accel_arm_x"         "accel_arm_y"         "accel_arm_z"
## [25] "magnet_arm_x"        "magnet_arm_y"        "magnet_arm_z"
## [28] "roll_dumbbell"       "pitch_dumbbell"      "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"   "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"    "accel_dumbbell_x"    "accel_dumbbell_y"
## [37] "accel_dumbbell_z"    "magnet_dumbbell_x"   "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"   "roll_forearm"        "pitch_forearm"
## [43] "yaw_forearm"         "total_accel_forearm" "gyros_forearm_x"
## [46] "gyros_forearm_y"     "gyros_forearm_z"     "accel_forearm_x"
## [49] "accel_forearm_y"     "accel_forearm_z"     "magnet_forearm_x"
## [52] "magnet_forearm_y"    "magnet_forearm_z"    "classe"
```

Build the prediction model

A random forest model is build:

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

```
set.seed(21469)
modelFit <- randomForest(trainData$classe~.,data=trainData)
```

We can now look at the prediction that the model gives for the training data (created as 70% randomly selected from the original test set):

```
predTrain <- predict(modelFit,newdata=trainData)
confusionMatrix(predTrain,trainData$classe)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 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
```

The 'in' sample error is 0% here which is of course an unrealistic estimate of the sample error. Therefore we will now test our model on the test set we build as 30% of our original test set:

```
predTest <- predict(modelFit,newdata=testData)
confusionMatrix(predTest,testData$classe)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    4    0
##          D    0    0    0  960    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  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   0.9959   0.9982
## Specificity            1.0000   1.0000   0.9992   0.9996   1.0000
## Pos Pred Value         1.0000   1.0000   0.9961   0.9979   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9992   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1631   0.1835
## Detection Prevalence   0.2845   0.1935   0.1750   0.1635   0.1835
## Balanced Accuracy      1.0000   1.0000   0.9996   0.9977   0.9991
```

We now have an out of sample accuracy of 99.8% which is a better estimate as before. Content with this accuracy we will now predict the outcomes of the real test set.

Predict on test set

Load the data and predict the 20 values:

```
pmltest <- read.csv('pml-testing.csv',na.strings=c("","NA","#DIV/0!"))
predictions <- predict(modelFit,newdata=pmltest)
predictions
```

```
##  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
```