Machine Learning Project

Steve Burch

Sunday, May 24, 2015

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, the 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 can be found here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Goal & Expectation

We are going to run our "trained" model against the 20 Test Cases provided.

I would like <= 1% error in my predictive model.

The Data

The training data for this project are available here:

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

And, the test data are available here:

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

Data Manipulation

(Please set your data directory appropriately)

```
setwd("E:/MyStuff/Myfiles/Coursera/8_MachineLearning")
library(caret)
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

read files previously downloaded files

```
training <- read.csv("pml-training.csv")
test_data <- read.csv("pml-testing.csv")
dim(training_data)
```

```
## [1] 13737    53
```

The training_data has 19,622 observations of 160 variables.

split the training data into training and validation data sets

```
set.seed(777)
trainingParition <- createDataPartition(training$classe, p = 0.7, list = FALSE)
training_data <- training[trainingParition, ]
validation_data <- training[-trainingParition, ]
```

We will exclude NAs and near zero variance features from the training data, and other nefarious columns

```
nearZeroCols <- nearZeroVar(training_data)
training_data <- training_data[, -nearZeroCols]
```

Find cols that have some NAs or are empty, we want to count occurrences

```
badColsLen <- sapply(training_data, function(x) {
    sum(!(is.na(x) | x == ""))
})
```

Get rid of those with 40% or more of missing values

```
tossCols <- names(badColsLen[badColsLen < 0.6 * length(training_data$classe)])
training_data  <- training_data[, !names(training_data) %in% tossCols]
```

And other Columns I don't want

```
tossCols <- c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","new_windo
training_data  <- training_data[, !names(training_data) %in% tossCols]

dim(training_data)
```

```
## [1] 13737    53
```

Now we have 53 columns!

Training the Model Using Random Forest

We are trying to come up with a Model to predict the classe variable.
Random Forest is a good out-of-the-box choice.

```
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.
```

```
modelRF <- randomForest(classe ~ ., data = training_data, importance = TRUE, ntrees = 10)
```

Test the Model

We will use the model against the training data first, to examine how well Random Forest worked on our reduced set of columns.

```
predictTraining <- predict(modelRF, training_data)
print(confusionMatrix(predictTraining, training_data$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
```

We are getting an accuracy of 100%, with a 95% CI, so that's excellent! (or we have over fitted)

Cross Validate against our Validation data set (which still has all the extra columns) to see how well the model works

```
predictVal <- predict(modelRF, validation_data)
print(confusionMatrix(predictVal, validation_data$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    0 1130    3    0    0
##          C    2    2 1023   10    3
##          D    0    0    0  953    4
##          E    0    0    0    1 1075
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9921   0.9971   0.9886   0.9935
## Specificity            0.9983   0.9994   0.9965   0.9992   0.9998
## Pos Pred Value         0.9958   0.9974   0.9837   0.9958   0.9991
## Neg Pred Value         0.9995   0.9981   0.9994   0.9978   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1920   0.1738   0.1619   0.1827
## Detection Prevalence   0.2853   0.1925   0.1767   0.1626   0.1828
## Balanced Accuracy      0.9986   0.9957   0.9968   0.9939   0.9967
```

Here we get .9946, so we can be happy with our RF Model.

Apply our Model to the Test data set

```
predictTest <- predict(modelRF, test_data)
##predictTest   commented out so I don't give answers away/get in trouble
```

Now, we want to write the output to files...

```
predictTest <- as.vector(predictTest)
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(predictTest)
```

THE END!