

Machine Learning Writeup, Data Science Specialization

Charles Carter*

October 17, 2014

Contents

1	Introduction	1
2	Instructions	1
2.0.1	Background	1
2.0.2	Data	2
2.0.3	What you should submit	2
2.0.4	Reproducibility	2
3	Helper Functions	3
3.0.5	Loading Packages	3
3.0.6	Helper Functions	3
3.0.7	Load Data	3
3.0.8	Convert Classe	3
3.0.9	Delete Empty Columns	4
4	Logic	4
4.0.10	Calling Helper Functions	4
4.0.11	Creating Training and Testing Data	5
4.0.12	Prediction	5
5	Results	6
6	Conclusion	6

*ccarter@troy.edu

1 Introduction

2 Instructions

2.0.1 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 dumbbell 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).

2.0.2 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.

2.0.3 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.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file¹ describing your analysis. Please constrain the text of the writeup to ≤ 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

¹I have not submitted Markdown or HTML files, but Rnw and PDF files. The reason is that I almost never use HTML output in my job, but frequently generate PDF files, so I wanted to focus on the production of output in PDF format.

2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

2.0.4 Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

3 Helper Functions

3.0.5 Loading Packages

My analysis uses a number of helper functions. This sections details these functions. The first order of business is to load the required packages. We need `caret` to perform the prediction and training, `knitr` to prepare the PDF, and `ggplot2` for visualization.

```
library(caret)
library(knitr)
library(ggplot2)
```

3.0.6 Helper Functions

Next, we load and clean the data. Function `load.data()` assumes that the data file is in the current directory. This function loads the data into the R environment, deletes the `user_name` column (which is not relevant to the analysis), and calls the `convert.classe()` function to convert the character outcome variable into integer values. I did this in order to check how closely the prediction results matched the outcome, which I could not do with discrete qualitative variables. Then, I deleted the columns with NA values and the empty character columns with `delete.empty.cols()`.

3.0.7 Load Data

```
load.data <- function()
{
  train <- read.csv("pml-training.csv", stringsAsFactors = FALSE)
  train$user_name <- NULL
  train$classeNum <- sapply(train$classe, convert.classe)
  print("training set loaded, returns 'train'")
  return(train)
}
```

3.0.8 Convert Classe

```
convert.classe <- function(x)
{
  if(x == "A") { return(1) }
  else if(x == "B") { return(2) }
  else if(x == "C") { return(3) }
  else if(x == "D") { return(4) }
  else if(x == "E") { return(5) }
  else { return(0) }
}
```

3.0.9 Delete Empty Columns

```
delete.empty.cols <- function(df)
{
  x <- train[, apply(train, 2, function(x) !any(is.na(x)))]
  x <- x[sapply(x, is.numeric)]
  print("training set cleaned and prepared, reeturns 'trainA'")
  return(x)
}
```

4 Logic

4.0.10 Calling Helper Functions

After creating the helper functions, I call them with the following listing. `train` consists of the original, raw data. `trainA` consists of the data after cleaning. This results in a model fit for the testing

```
#this is the program logic
print("calling 'train <- load.data()'")

## [1] "calling 'train <- load.data()'"

train <- load.data()

## [1] "training set loaded, returns 'train'"

dim(train)

## [1] 19622 160

print("calling trainA <- delete.empty.cols(train)")
```

```
## [1] "calling trainA <- delete.empty.cols(train)"

trainA <- delete.empty.cols(train)

## [1] "training set cleaned and prepared, reeturns 'trainA'"

dim(trainA)

## [1] 19622    57
```

4.0.11 Creating Training and Testing Data

```
print("creating training and testing sets")

## [1] "creating training and testing sets"

inTrain <- createDataPartition(y = trainA$classeNum, p = 0.75, list = FALSE)
training <- trainA[inTrain, ]
testing <- trainA[-inTrain, ]
cat("Dimensions of training data are: ", dim(training), " and dimensions of the testing data are: ", dim(testing))

## Dimensions of training data are: 14718 57 and dimensions of the testing data are: 4904 57

print("setting seed and calling train()")

## [1] "setting seed and calling train()"

set.seed(141017)

modelFit <- train(classeNum ~ ., data = training, method = "glm")
```

4.0.12 Prediction

Finally, we run a prediction on the testing data set and observe the results. `predictions` contains the predictions of the testing set from the model trained in the training set. `result` contains the numerical results of the predictions. We then plot the results, as contained in the testing set, with the predictions, as shown in the plot below.

```
predictions <- predict(modelFit, newdata = testing)
result <- data.frame(cbind(testing$classeNum, predictions))
#look at the beginning and the end of the result file
head(result)

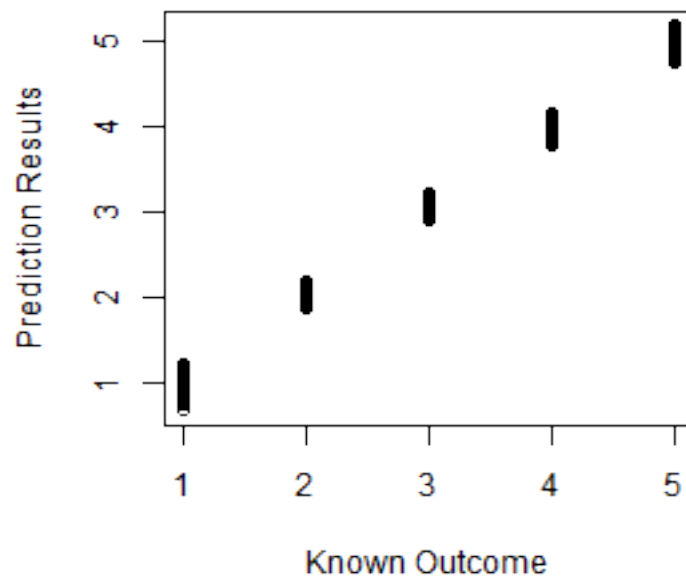
##      V1 predictions
```

```
## 1 1 0.9524
## 2 1 0.9475
## 3 1 0.9542
## 4 1 0.9500
## 5 1 0.9506
## 6 1 0.9495

tail(result)

##      V1 predictions
## 4899 5      4.860
## 4900 5      4.873
## 4901 5      4.887
## 4902 5      4.874
## 4903 5      4.869
## 4904 5      4.892

plot(result1$V1, result1$predictions, xlab = 'Known Outcome', ylab = 'Prediction Results')
```



Both variables, **predictions** and **V1** (the known outcome variable) are coded as numeric variables. The plot shows an absolute separation between the groups of outcome variables. As shown by the output from **head()** and **tail()** above,

the predictions are very close to the integer variables of the outcomes.

5 Results

6 Conclusion