Course 8 Project

iabalki

Monday, November 21, 2016

The Assignment

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. [...] 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.

Goal

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.

Data Input - Sources

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
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 Input Explained

The information in this section is sourced from the above "more information" page. The objective of the Weight Lifting Exercises dataset is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training.

The owners of the experiement used (among other approaches) an on-body sensing approach (dataset here), where they attached sensors to six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)

- lowering the dumbbell only halfway (Class D) and
- throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Requirements for completion of the Assignment:

- 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
- Use your prediction model to predict 20 different test cases [see summary]
- Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. [Total words: 1996; 1 figure]
- 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 [note: submitting as PDF so that the text is formatted reasonably and HTML where I have no control over format and readability]

Legal Disclaimer

This data was only used for a Coursera machine learning class. The data was obtained from the following study: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Summary of Completed Analysis on this Assignment

After the data was loaded and cleaned (columns and rows removed) - two algorims were used to predict the values of a testing dataset [part of the training dataset] - decision trees (accuracy 77%) and random forests (accuracy 99%). The random forest method was selected and predictions were created with it, as follows:

Case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Prediction	B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B

Step-By-Step Execution incl Design Choices

The following section will describe the how the model was build and why certain choices were made. All the relevant code is also included in order to make the results reproducable.

Load the Data

The data was loaded in memory:

```
training <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), na.strings=c("NA","#DIV/0!",""))

#in order to reproduce it
set.seed(12345)
setwd ("D:/Users/Milla/Documents/Data Science Classes/Course 8 Project")
```

Explore the data

Here I poked around the data to get a look and feel of it using the folowing commangs (execution is omitted to fit in the size limits)

```
## poke around the data to see it..
str(training)
str(testing)
summary (testing)
summary (training)
View (training)
View(testing)
```

Then I had to ask and answer some more specific questions:

```
## is X a unique identifier (kinda the same as row number)?
length (unique (training$X))==nrow (training)

## [1] TRUE

output <- NULL
for (i in 1:nrow (training)) {if (!(training$X [i]==i)) {output [i] == FALSE}
else {output [i] = TRUE}}
summary(output)

##    Mode    TRUE    NA's
## logical   19622       0

## seems like it :)
```

```r
## check which columns that are in training are not in test
for (i in 1:ncol (training)) {if (!(names(training)[i] %in% names (testing)))
{print(names(training)[i])}}
```

```
## [1] "classe"
```

```r
## check which columns that are in testing are not in training
for (i in 1:ncol (testing)) {if (!(names(testing)[i] %in% names (training)))
{print(names(testing)[i])}}
```

```
## [1] "problem_id"
```

Observations from the exploration:

- There is no data documentetion on all of the variables describing their meaning [and no data analyst should do analysis on data they don't understand in in real life :)]
- The training dataset has 19622 observations and 160 variables, and the testing data set has 20 observations and the same number of variables as the training set, however:
    - We are trying to predict the outcome of the variable "classe" (a factor variable with possible options A-E) which is in training but not in testing
    - The testing set seems to have an addititional variable called "problem_id" - presumable it will be needed for identifying the use case for the predicted output.

- While there are 160 variables (columns) in the training set -- a lot of those seem to be a summary statsitics for a single exercise performed by an individiual -- those only contain values where "new_window" variable == "yes".

Therefore before we start we should:

- remove all summary lines i.e. rowns where "new_window" variable == "yes"
- remove all columns that are empty threafter (we will load blanks as NAs to more easily identify those)
- remove column X as it is an identifier and we do not want to include it in the model
- remove the number window (two diff columns by the name "new_window" as they are also identifiers for the use case and should not be used for predicting
- remove - timestamps -- timestamps might be good predictors if the subjects were part of an experiment where their endurance was tested but given that this was a light weight and how-well-did-we-perform-the-exercise test, the timestamps would likley not add significant value (besides for chronological purposes). In addition, they could have helped with calulating the motion (acceleration / speed, etc) but since we neither understand the data, nor we have understnading of basic physics and how these data points tie together.. for the purpose of this assignement - I will ignore them.
- remove the identifier of the person who is performing the exercise (not something you can control for future predictions)

Clean the data

Now, let's execute the plan made ealier.

```
## there are two new_window columsn and we can't use them until we name them
differntly
names (training) [6] <- "new_window_fac"
## remove any line item with summary values
trainClean<-subset(training, new_window_fac=="no")
## remove any columns where ther are more than 50% of NAs in the column
(likely to be poor predictor)
excludeLevel <- nrow (trainClean)*0.5
output<-NULL
for (i in 1:ncol (trainClean)) {if (sum(is.na(trainClean[, i])) > excludeLeve
l) {output <-c(output,i)}}
trainClean <-trainClean[,-output]
rm(excludeLevel)

## remove the fist 7 columns that contain X, user name, new windwows id and t
he timestamps - see reasoning in the observations in previous section
trainClean<-trainClean [,-c(1:7)]
```

Create a training and testing set

```
library (caret)

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

inTrain <- createDataPartition(trainClean$classe, p=0.6, list=FALSE)
myTrain <- trainClean[inTrain, ]
myTest <- trainClean[-inTrain, ]
dim(myTrain); dim(myTest)

## [1] 11532    53

## [1] 7684   53
```

Predicting with Decision Trees + Cross-Validation

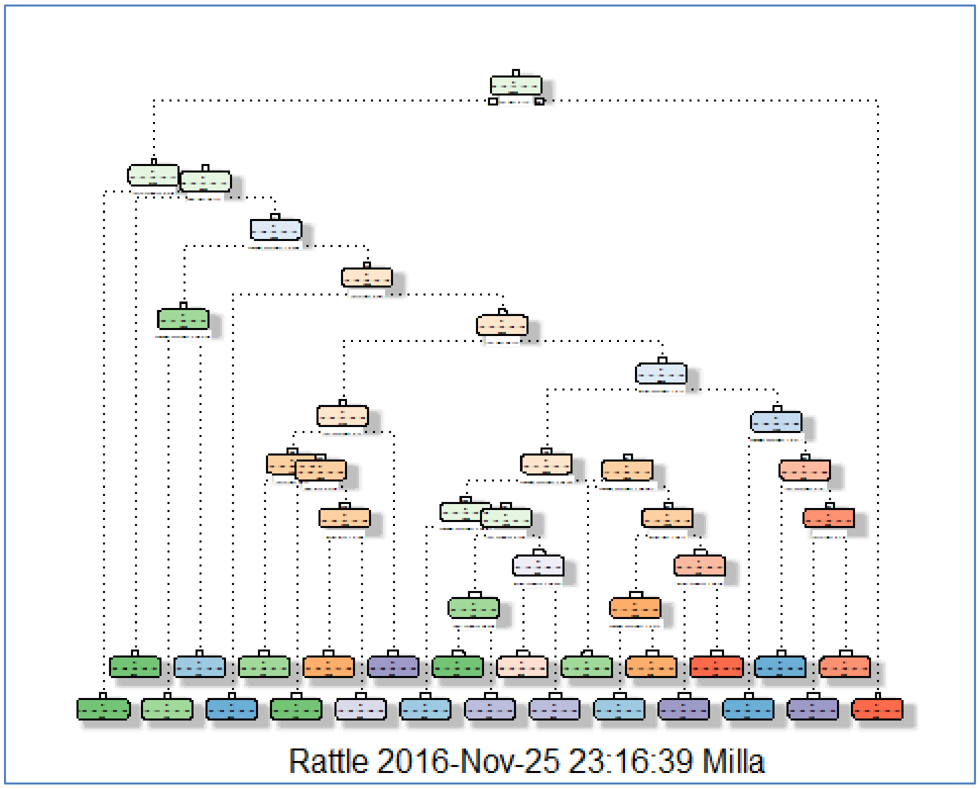
Here we will let R create a decision tree model that will allow us to predice "classe". The resulting predicted value will be compared to the actual known value in my test dataset and accuracy of the prediction will be evaluated.

```
library(AppliedPredictiveModeling)
library (rpart)

suppressMessages(library(rattle))
library(rpart.plot)
```

```
model_dt <- rpart(classe ~ ., data=myTrain, method="class")
fancyRpartPlot(model_dt)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
prediction_dt <- predict(model_dt, myTest, type = "class")
confusionMatrix(prediction_dt, myTest$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1951  242   45   46   21
##          B   27  973   83   68   88
##          C   71   88 1031  209   69
##          D   70   53   94  861   80
##          E   69  131   87   74 1153
##
## Overall Statistics
##
##                Accuracy : 0.7768
##                  95% CI : (0.7673, 0.7861)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7172
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8917   0.6543   0.7694   0.6844   0.8172
## Specificity            0.9356   0.9571   0.9311   0.9538   0.9425
## Pos Pred Value         0.8464   0.7853   0.7023   0.7435   0.7616
## Neg Pred Value         0.9559   0.9202   0.9503   0.9392   0.9582
## Prevalence             0.2847   0.1935   0.1744   0.1637   0.1836
## Detection Rate         0.2539   0.1266   0.1342   0.1121   0.1501
## Detection Prevalence   0.3000   0.1612   0.1910   0.1507   0.1970
## Balanced Accuracy      0.9136   0.8057   0.8503   0.8191   0.8798
```

Accuracy for this 77% which is relatively low.

Predicting with Random Forest + Cross Validation

Here we will let R create a model using the random forest method that will allow us to predict "classe". The resulting predicted value will be compared to the actual known value in my test dataset and accuracy of the prediction will be evaluated.

```
suppressMessages(library(randomForest))
model_rf <- randomForest(classe ~ ., data=myTrain)
prediction_rf <- predict(model_rf, myTest, type = "class")
confusionMatrix(prediction_rf, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2184    8    0    0    0
##          B    0 1472    5    0    0
##          C    3    7 1332    4    2
##          D    0    0    3 1254    8
##          E    1    0    0    0 1401
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9928, 0.9962)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9899   0.9940   0.9968   0.9929
## Specificity            0.9985   0.9992   0.9975   0.9983   0.9998
## Pos Pred Value         0.9964   0.9966   0.9881   0.9913   0.9993
## Neg Pred Value         0.9993   0.9976   0.9987   0.9994   0.9984
## Prevalence             0.2847   0.1935   0.1744   0.1637   0.1836
## Detection Rate         0.2842   0.1916   0.1733   0.1632   0.1823
## Detection Prevalence   0.2853   0.1922   0.1754   0.1646   0.1825
## Balanced Accuracy      0.9984   0.9946   0.9958   0.9976   0.9964
```

Accuracy is 99%.

Selecting the Model

Now given that accuracy is our main criteria, I elected to use the random forest model and predict the values for original testing set:

- Decision Trees – Accuracy 77%
- Random Forest – Accuracy 99%

Predicting the values

```
prediction <- predict(model_rf, testing, type = "class")
prediction
```

```
##  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
```

the end