Project on Practical Machine Learning

sougata biswas

Monday, September 22, 2014

Practical Machine Learning - Prediction Assignment Writeup

For this assignment I analyzed the provided data to determine what activity an individual perform. To do this I made use of caret and randomForest, this allowed me to generate correct answers for each of the 20 test data cases provided in this assignment. I made use of a seed value for consistent results.

```
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(caret)
```

```
## Loading required package: ggplot2
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:survival':
##
##     cluster
```

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Hmisc':
##
##     combine
```

```
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators
## Loading required package: parallel
```

```
set.seed(2048)
options(warn=-1)
```

First, I loaded the data both from the provided training and test data provided by COURSERA. Some values contained a "#DIV/0!" that I replaced with an NA value.

```
training_data <- read.csv("pml-training.csv", na.strings=c("#DIV/0!") )
evaluation_data <- read.csv("pml-testing.csv", na.strings=c("#DIV/0!") )
```

I also casted all columns 8 to the end to be numeric.

```
for(i in c(8:ncol(training_data)-1)) {training_data[,i] = as.numeric(as.character(training_data[,i]))}

for(i in c(8:ncol(evaluation_data)-1)) {evaluation_data[,i] = as.numeric(as.character(evaluation_data[,
```

Some columns were mostly blank. These did not contribute well to the prediction. I chose a feature set that only included complete columns. We also remove user name, timestamps and windows.

Determine and display out feature set.

```
feature_set <- colnames(training_data[colSums(is.na(training_data)) == 0])[-(1:7)]
model_data <- training_data[feature_set]
feature_set
```

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

We now have the model data built from our feature set.

```
idx <- createDataPartition(y=model_data$classe, p=0.75, list=FALSE )
training <- model_data[idx,]
testing <- model_data[-idx,]
```

We now build 5 random forests with 150 trees each. We make use of parallel processing to build this model. I found several examples of how to perform parallel processing with random forests in R, this provided a great speedup.

```
registerDoParallel()
x <- training[-ncol(training)]
y <- training$classe

rf <- foreach(ntree=rep(150, 6), .combine=randomForest::combine, .packages='randomForest') %dopar% {
randomForest(x, y, ntree=ntree)
}
```

Provide error reports for both training and test data.

```
predictions1 <- predict(rf, newdata=training)
confusionMatrix(predictions1,training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

```
predictions2 <- predict(rf, newdata=testing)
confusionMatrix(predictions2,testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  941    6    0    0
##          C    0    3  848    8    2
##          D    0    0    1  796    3
##          E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.996)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.992    0.992    0.990    0.994
## Specificity             0.999    0.998    0.997    0.999    1.000
## Pos Pred Value          0.996    0.994    0.985    0.995    1.000
## Neg Pred Value          1.000    0.998    0.998    0.998    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.162    0.183
## Detection Prevalence    0.285    0.193    0.176    0.163    0.183
## Balanced Accuracy       0.999    0.995    0.994    0.995    0.997
```

Conclusions and Test Data Submit

As can be seen from the confusion matrix this model is very accurate. I did experiment with PCA and other models, but did not get as good of accuracy. Because my test data was around 99% accurate I expected nearly all of the submitted test cases to be correct. It turned out they were all correct.

Prepare the submission. (using COURSERA provided code)

```
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)
  }
}


x <- evaluation_data
x <- x[feature_set[feature_set!='classe']]
answers <- predict(rf, newdata=x)

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
```

```
pml_write_files(answers)
```