Practical Machine Learning: Course Project

mghaz007

May 19th, 2015

Prediction of Correct/Incorrect Barbell Lifting

1. Executive Summary

In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants in order to predict whether they perform barbell lifts correctly or incorrectly. In the experiment, the participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Whethe ror not the exercise was performed correctly is the "class" variable in the training set. We explored the use of various explanatory attributes to predict class variable. This report describes how we built our prediction model, how we applied cross validation, our assessment of the expected out of sample error is, and explaination the implemented choices. In the end, we also applied our prediction model to predict 20 different test cases and siubmitted the results.

2. Machine Learning Algorithms

For this assignment, we analyzed the provided data to predict whether or not an individual has performed barbell lifts correctly. To accomplish this, we applied two machine learning algorithms: caret and randomForest algorithms. These two algorithms yield perfect predictions and correct answers for each of the 20 test data cases provided in this assignment. In order to be able to reproduce the results, an initial seed for the random number generator was set.

```
library(Hmisc)
library(caret)
library(randomForest)
library(foreach)
library(doParallel)
set.seed(2048)
options(warn=-1)
```

3. Reading and Preparing and Cleaning the Data

The provided training and test data sets were first read. We conducted an initial data preparation and cleaning of the input data, which involved the following operations:

- Some values contained a "#DIV/0!", which were replaced with an NA value, as follows:

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

- We also casted all columns from the 8th column to the end to be numeric, as follows:

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

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

- We ensured that some columns, which are mostly blank did not contribute to the prediction, by selecting a feature set that only included complete columns.
- We also removed user name, timestamps and windows.

Feature Selection We selected a feature set that only included complete columns:

- Determine and display out feature set, as follows:

```
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 generate the model data from our feature set, we follows:

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

Apply Machine Learning Algorithms We 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
randomForest(x, y, ntree=ntree)
}
```

Prediction Results

We also provide prediction 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 : (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   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  942    6    0    0
##          C    0    3  848    8    2
##          D    0    0    1  796    2
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                 Accuracy : 0.9947
##                   95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     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            1.0000   0.9926   0.9918   0.9900   0.9956
## Specificity            0.9989   0.9985   0.9968   0.9993   1.0000
## Pos Pred Value         0.9971   0.9937   0.9849   0.9962   1.0000
## Neg Pred Value         1.0000   0.9982   0.9983   0.9981   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1729   0.1623   0.1829
## Detection Prevalence   0.2853   0.1933   0.1756   0.1629   0.1829
## Balanced Accuracy      0.9994   0.9956   0.9943   0.9947   0.9978
```

Test Cases Submission

This project requires the submission of the results of 20 test cases. The codes for generating the output for these test cases is as follows:

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("./output/test_cases_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)
```

Conclusions

As can be seen from the prediction performance measures, such as the confusion matrix, the implmented model is very accurate. For test data, we expected to get 99% accuracy, but in fact we achieved a 100% accuracy, where in fact correct all test cases were predicted correctly. We shold not that we explored various prediction models, such as the PCA, but we did not get as good of accuracy.