Practical Machine Learning : Course Projects

W Wocher

April 24, 2016

Executive Summary

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. The goal of your project is to predict the manner in which the 6 participants did the exercise from accelerometers on the
- belt
- forearm
- arm
- dumbell

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

Synopsis

Create 2 sets of data from the training data.

1 Try several different models and compare how accurate each of them are.
2 Once a model is decided on, then run predictions on the training data based on splitting is up into 2 data sets.
3 Load the data from the test file
4 Apply the model to this test file

Exploratory Data Analysis

Install Libraries and Set Seed

```
library(lattice)
library(ggplot2)
library(mlbench)
library(caret)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
```

```
##
##     margin

library(foreach)
library(iterators)
library(parallel)
library(doParallel)
set.seed(64)
```

Load Data

```
setwd("D:/Coursera/08 Practical Machine Learning/Project")
training.pml <- read.csv(file="pml-training.csv",head=TRUE,sep=",")
testing.pml <- read.csv(file="pml-testing.csv",head=TRUE,sep=",")
```

Eliminate columns

Many of the columns do not have a lot of data. Remove columns that have NAs in them.

```
training.complete <- training.pml[sapply(training.pml, function(x) !any(is.na(x)))]
```

Columns for analysis

The goal is to use data from accelerometers on the belt, forearm, arm, and dumbell.

```
column_names <- c("classe",
                  grep("belt", colnames(training.complete), value = TRUE)
                , grep("forearm", colnames(training.complete), value = TRUE)
                , grep("arm", colnames(training.complete), value = TRUE)
                , grep("dumbell", colnames(training.complete), value = TRUE)
                 )
column_names <- unique(column_names)
```

Subset the data based on the columns

The goal is to use data from accelerometers on the belt, forearm, arm, and dumbell.

```
training.subset <- subset(training.complete, select=column_names)
```

The data set currently has 64 columns. This includes the outcome and the predictors.
Modeling can not handle categorical predictors with more than 53 categories.
The following steps is meant to reduce the number of columns.

Convert the columns to numeric.

This step introduces NAs. Eliminate all the colunns with NAs.

```
for(i in 2:ncol(training.subset)) {
    training.subset[,i] = as.numeric(as.character(training.subset[,i]))
}
training.subset <- training.subset[sapply(training.subset, function(x) !any(is.na(x)))]
```

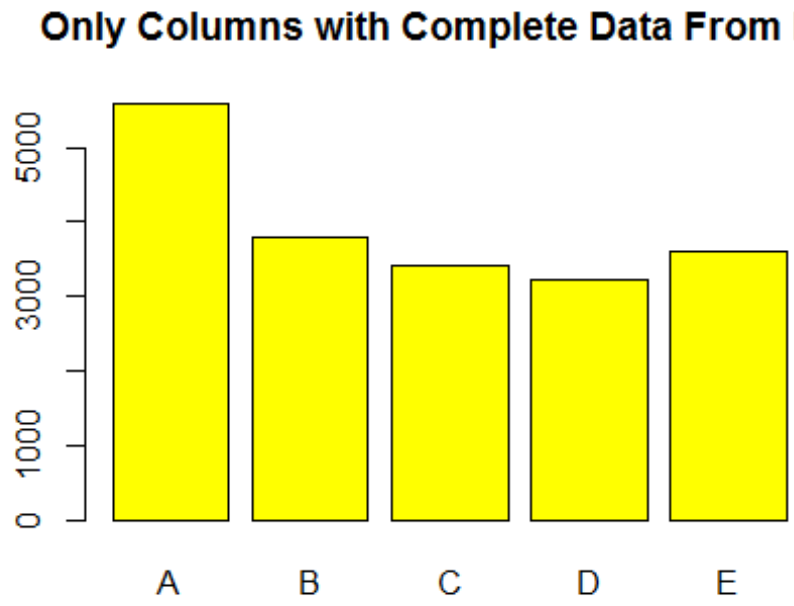
The data set currently has 40 columns.

Plot

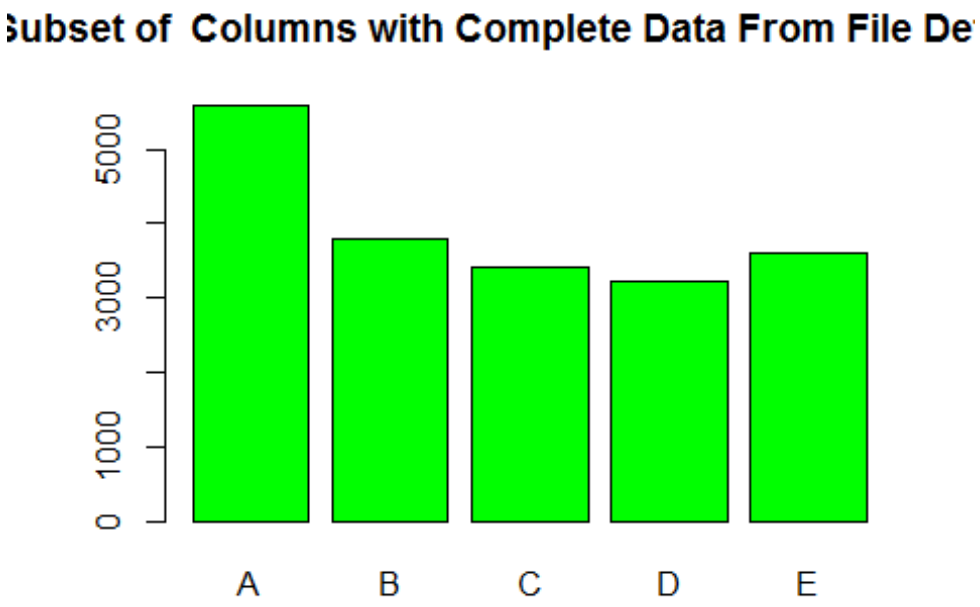
Plot the different data sets to see if anything has drastically changed

```
par(mfrow=c(1,1))
plot(training.pml$classe, col="blue", main="All Data From File")
```


```
plot(training.complete$classe, col="yellow", main="Only Columns with Complete Data From F
ile")
```




```
plot(training.subset$classe, col="green", main="Subset of  Columns with Complete Data Fro
m File Defined")
```




```
par(mfrow=c(1,1))
```

Modeling

Split the data

```
inTrain <- createDataPartition(y=training.subset$classe, p=0.7, list=FALSE)
training <- training.subset[inTrain,]
testing <- training.subset[-inTrain,]
```

Setup parallel processing

Configure the trainControl object to be used in the modeling.

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv",
                           number = 10,
                           allowParallel = TRUE)
```

Develop training model based on rf

```
training.fit.rf <- train(classe ~ ., method="rf", data=training, trControl = fitControl)
```

Develop training model based on knn

```
training.fit.knn <- train(classe ~ ., method="knn", data=training, trControl = fitControl
)
```

Develop training model based on rpart

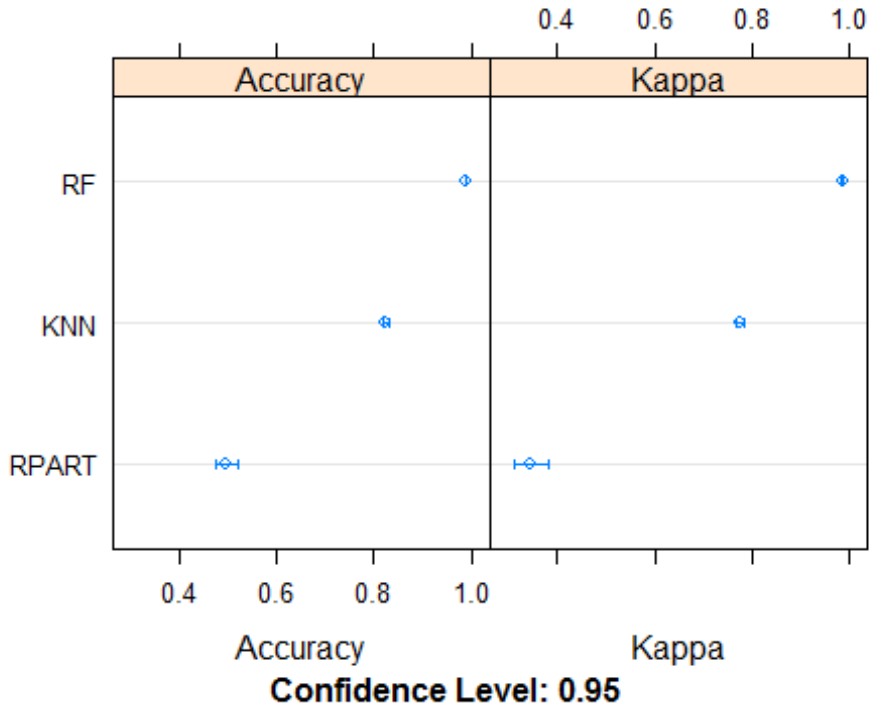
```
library(rpart)
training.fit.rpart <- train(classe ~ ., method="rpart", data=training, trControl = fitCon
trol)
```

Collect resamples and compare models

```
results <- resamples(list(RF=training.fit.rf, KNN=training.fit.knn, RPART=training.fit.rp
art))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: RF, KNN, RPART
## Number of resamples: 10
##
## Accuracy
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## RF    0.9869  0.9884 0.9898 0.9900  0.9913 0.9942    0
## KNN   0.8093  0.8177 0.8242 0.8241  0.8326 0.8362    0
## RPART 0.4618  0.4686 0.4867 0.4955  0.5230 0.5440    0
##
## Kappa
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## RF    0.9834  0.9853 0.9871 0.9874  0.9890 0.9926    0
## KNN   0.7589  0.7692 0.7776 0.7775  0.7883 0.7926    0
## RPART 0.2986  0.3026 0.3254 0.3451  0.3903 0.4235    0
```

```
par(mfrow=c(1,1))
dotplot(results)
```



After comparing the models, Random Forest has the best results.

Predicting

Prediction of the Training File : Training Set

```
training.prediction.rf <- predict(training.fit.rf$finalModel, training, type = "class")
confusionMatrix(training$classe,training.prediction.rf)
```

```
## 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
```

Prediction of the Training File : Test Set

```
testing.prediction.rf <- predict(training.fit.rf$finalModel, testing, type = "class")
confusionMatrix(testing$classe,testing.prediction.rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    1    0
##          B    1 1136    2    0    0
##          C    0   11 1004   11    0
##          D    0    0   22  939    3
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.9882, 0.9932)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9895   0.9767   0.9863   0.9972
## Specificity            0.9995   0.9994   0.9955   0.9949   0.9998
## Pos Pred Value         0.9988   0.9974   0.9786   0.9741   0.9991
## Neg Pred Value         0.9998   0.9975   0.9951   0.9974   0.9994
## Prevalence             0.2843   0.1951   0.1747   0.1618   0.1842
## Detection Rate         0.2841   0.1930   0.1706   0.1596   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9995   0.9945   0.9861   0.9906   0.9985
```

Prediction of the Testing File

Load the Testing file and apply the model

```
testing.pml.prediction.rf <- predict(training.fit.rf, testing.pml)

testing.pml.prediction.rf
```

```
##  [1] 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
```

De-register parallel processing cluster

```
stopCluster(cluster)
```