Practical Machine Mearning Project

Constantinos Lirigos

April 2017

```
library(caret)
library(rpart)
library(randomForest)
library(nnet)
library(MASS)
set.seed(1234)
```

Summary

The aim of the project is to analyze data from accelerometers on 6 participants. These participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways and graded by experts (grades A, B, C, D, E). The data collected will be used to build a model in order to be able to predict if an exercise was performed correctly according to accelerometer readings. We downloaded a training set with 19622 observations of 160 variables (for model building) and a testing set with only 20 observations with the same 160 variables.

Getting Data

```
url_1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
url_2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
download.file(url_1, "training.csv")
download.file(url_2, "testing.csv")
training <- read.csv("training.csv")
dim(training)

## [1] 19622   160

testing <- read.csv("testing.csv")
dim(testing)

## [1]  20 160
```

Partitioning

We will partition the "training" set in two sub-sets: train with 70% of the data (to build our models) and "test" with 30% (to test the models). Then we will use the best model to predict on the "testing" data.

```
inTrain <- createDataPartition(training$classe, p = 0.7)[[1]]
train <- training[ inTrain,]
test <- training[-inTrain,]
```

Cleaning Data

We will remove the collumns with NAs, as well as the collumns (variables) with very low variance.

```
na_count <-sapply(train, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
na <- which(na_count[1] > 13000)
train <- train[, -na]
dim(train)

## [1] 13737    93

a <- nearZeroVar(train)
train <- train[, -a]
dim(train)

## [1] 13737    59
```

Then we will remove variables that do not contribute anything to the analysis, such as user name, time stamps, etc, i.e variables "1" to "7"

```
train <- train[, -c(1:7)]
dim(train)

## [1] 13737    52
```

We will coerce all variables to numeric:

```
train[,c(1:51)] <- sapply(train[,c(1:51)], as.numeric)
```

We have reduced the variables from 160 to just 52. Then we will keep the same collumns in the test set and also coerce all variables to numeric, as we did in the train set.

```
test<- test[,colnames(train)]
test[,c(1:51)] <- sapply(test[,c(1:51)], as.numeric)
```

Model Building

We will try the following models: decision trees (rpart), random forests (rf), Generalized linear model (glm) and Linear Discriminant Analysis (lda). We will not present the whole confusion matrix, but only the respective accuracies.

Decision tree (rpart)

```
fit_rpart <- rpart(classe ~ ., data = train)
predict_rpart <- predict(fit_rpart, newdata = test, type = "class")
```

```
rpart <- confusionMatrix(predict_rpart,
test$classe)$overall["Accuracy"]
rpart

## Accuracy
## 0.728972
```

Random Forests (rf)

```
fit_rf <- randomForest(classe ~., data = train, ntree = 30)
predict_rf <- predict(fit_rf, newdata = test)
rf <- confusionMatrix(predict_rf, test$classe)$overall["Accuracy"]
rf

##  Accuracy
## 0.9960918
```

Generalized Linear Model (glm)

Because the outcome is not binary and has 5 levels, we will use the "nnet" package

```
fit_glm <- multinom(classe ~ ., data = train)

## # weights:  265 (208 variable)
## initial  value 22108.848603
## iter  10 value 17650.893721
## iter  20 value 15766.998016
## iter  30 value 14598.224541
## iter  40 value 13917.931730
## iter  50 value 13371.236511
## iter  60 value 13058.082115
## iter  70 value 12841.763341
## iter  80 value 12666.829680
## iter  90 value 12600.278605
## iter 100 value 12540.261896
## final  value 12540.261896
## stopped after 100 iterations

predict_glm <- predict(fit_glm, newdata = test)
glm <- confusionMatrix(predict_glm, test$classe)$overall["Accuracy"]
glm

## Accuracy
## 0.664401
```

Linear Discriminant Analysis (lda)

```
fit_lda <- train(classe ~., method = "lda", data = train)
predict_lda <- predict(fit_lda, newdata = test)
lda <- confusionMatrix(predict_lda, test$classe)$overall["Accuracy"]
lda
```

```
##  Accuracy
## 0.6937978
```

To summarize things, we have examined 4 models: "rpart", "rf", "glm" and "lda" and obtained the following accuracies respectively:

```
rpart

## Accuracy
## 0.728972

rf

##  Accuracy
## 0.9960918

glm

## Accuracy
## 0.664401

lda

##  Accuracy
## 0.6937978
```

The model with the best accuracy is Random Forests with an accuracy of 0.9918437. The out of sample error rate is 0.0081563 < 1%.

Prediction on Test Data

We will apply the Random Forrest fited model on the "testing" set.

```
predict(fit_rf, newdata = testing)

##  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
```