Practical\_Machine\_Learning

Charalampos Karyotis

8 July 2017

### Practical Machine Learning

#### Loading required libraries

library(lattice)  
library(ggplot2)  
library(caret)  
library(rpart)  
library(rattle)

## Rattle: A free graphical interface for data mining with R.  
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

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

#### Loading the training set. Working directory must be set approprietly

training<-read.csv("pml-training.csv")

##### You can see some details concerning the training set by using the following commands

##### dim(training)

##### head(training)

##### summary(training)

### Create data partition 70% training and 30% testing

#### The testing set provided will be used for validating the model

set.seed(333);  
inTrain<-createDataPartition(training$classe,p=0.70,list=FALSE)  
newtraining<-training[inTrain,]   
newtesting<-training[-inTrain,]

### Data Preprocessing

#### Preprocessing includes removing features with high NA percentage

#### Remove features with zero variability

#### Normalize

#### Remove features that are irrelevant such as ,counters, timestampts,dates etc...

#### There were a lot of features and samples so reducing the original number

#### and cleaning the dat would be very beneficial.

#### See the percentage of NA values for every feature

NApercentages<-apply(newtraining, 2, function(col)sum(is.na(col))/length(col))  
which(NApercentages>0.90) ##see the features that are to be excluded from the analysis

## max\_roll\_belt max\_picth\_belt min\_roll\_belt   
## 18 19 21   
## min\_pitch\_belt amplitude\_roll\_belt amplitude\_pitch\_belt   
## 22 24 25   
## var\_total\_accel\_belt avg\_roll\_belt stddev\_roll\_belt   
## 27 28 29   
## var\_roll\_belt avg\_pitch\_belt stddev\_pitch\_belt   
## 30 31 32   
## var\_pitch\_belt avg\_yaw\_belt stddev\_yaw\_belt   
## 33 34 35   
## var\_yaw\_belt var\_accel\_arm avg\_roll\_arm   
## 36 50 51   
## stddev\_roll\_arm var\_roll\_arm avg\_pitch\_arm   
## 52 53 54   
## stddev\_pitch\_arm var\_pitch\_arm avg\_yaw\_arm   
## 55 56 57   
## stddev\_yaw\_arm var\_yaw\_arm max\_roll\_arm   
## 58 59 75   
## max\_picth\_arm max\_yaw\_arm min\_roll\_arm   
## 76 77 78   
## min\_pitch\_arm min\_yaw\_arm amplitude\_roll\_arm   
## 79 80 81   
## amplitude\_pitch\_arm amplitude\_yaw\_arm max\_roll\_dumbbell   
## 82 83 93   
## max\_picth\_dumbbell min\_roll\_dumbbell min\_pitch\_dumbbell   
## 94 96 97   
## amplitude\_roll\_dumbbell amplitude\_pitch\_dumbbell var\_accel\_dumbbell   
## 99 100 103   
## avg\_roll\_dumbbell stddev\_roll\_dumbbell var\_roll\_dumbbell   
## 104 105 106   
## avg\_pitch\_dumbbell stddev\_pitch\_dumbbell var\_pitch\_dumbbell   
## 107 108 109   
## avg\_yaw\_dumbbell stddev\_yaw\_dumbbell var\_yaw\_dumbbell   
## 110 111 112   
## max\_roll\_forearm max\_picth\_forearm min\_roll\_forearm   
## 131 132 134   
## min\_pitch\_forearm amplitude\_roll\_forearm amplitude\_pitch\_forearm   
## 135 137 138   
## var\_accel\_forearm avg\_roll\_forearm stddev\_roll\_forearm   
## 141 142 143   
## var\_roll\_forearm avg\_pitch\_forearm stddev\_pitch\_forearm   
## 144 145 146   
## var\_pitch\_forearm avg\_yaw\_forearm stddev\_yaw\_forearm   
## 147 148 149   
## var\_yaw\_forearm   
## 150

#### Remove features with NA percentage more than 90%

newtraining <- subset(newtraining, select = -c(which(NApercentages>0.90)) )

#### Removefeatures with non zero variability

nsc<-nearZeroVar(newtraining,saveMetrics = TRUE)   
newtraining <- newtraining[,nsc$nzv==FALSE]

#### See what variables are categorical (except classe)

catvar<-sapply(newtraining,class)

##### Use the following command to explore user\_name variable

##### table(newtraining$user\_name)

#### Remove all irrelevant inputs

##### Remove the first column which is a count variable and the date and time columns

newtraining<-newtraining[,-c(1,3,4,5,6)]

#### Normalize

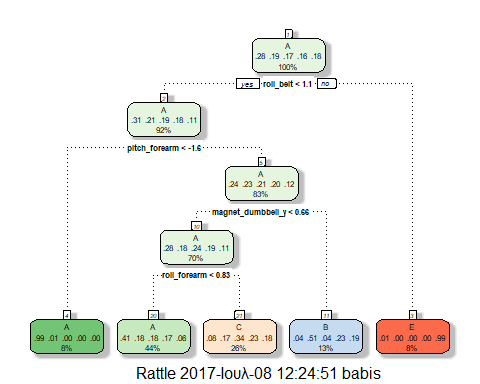
preobj<-preProcess(newtraining[,-54],method = c("center","scale"))  
newtraining1<-predict(preobj,newtraining[,-54])   
newtraining<-cbind(newtraining1,newtraining[,54])  
colnames(newtraining)[54] <- "classe"

#### Train a classification tree

set.seed(333);  
modelFit<-train(classe~.,method="rpart",data=newtraining)

##### Tree structure

fancyRpartPlot(modelFit$finalModel)



#### Train random forest (Takes a lot of time aprox 12 hours..... please use with caution)

##### I have saved in my working directory a trained model so that I can run this script without retraining....

##### If yu want to train the model use the following commands

##### set.seed(333);

##### modelFit2<-train(classe~.,method="rf",data=newtraining,prox=TRUE)

load("my\_model1.rda")

#### Tranform testing set using the same steps as with the training set

usedfeatures <- colnames(newtraining)  
newtesting<-newtesting[usedfeatures]  
dim(newtesting)

## [1] 5885 54

newtesting1<-predict(preobj,newtesting[,-54])  
newtesting<-cbind(newtesting1,newtesting$classe)  
colnames(newtesting)[54] <- "classe"

#### Calculate accuracy using testing set

#### Random Forest accuracy

predictions2<-predict(modelFit2,newdata=newtraining)  
CM22<-confusionMatrix(predictions2,newtraining$classe)  
CM22

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 3902 15 0 0 0  
## B 1 2603 31 18 0  
## C 2 19 2295 32 0  
## D 0 5 28 2200 12  
## E 1 16 42 2 2513  
##   
## Overall Statistics  
##   
## Accuracy : 0.9837   
## 95% CI : (0.9814, 0.9857)  
## No Information Rate : 0.2843   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9794   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9990 0.9793 0.9578 0.9769 0.9952  
## Specificity 0.9985 0.9955 0.9953 0.9961 0.9946  
## Pos Pred Value 0.9962 0.9812 0.9774 0.9800 0.9763  
## Neg Pred Value 0.9996 0.9950 0.9911 0.9955 0.9989  
## Prevalence 0.2843 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2841 0.1895 0.1671 0.1602 0.1829  
## Detection Prevalence 0.2851 0.1931 0.1709 0.1634 0.1874  
## Balanced Accuracy 0.9987 0.9874 0.9766 0.9865 0.9949

#### Simple classification tree accuracy

predictions<-predict(modelFit,newdata=newtesting)  
CM2<-confusionMatrix(predictions,newtesting$classe)  
CM2

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1524 463 488 440 147  
## B 27 376 33 169 156  
## C 119 300 505 355 302  
## D 0 0 0 0 0  
## E 4 0 0 0 477  
##   
## Overall Statistics  
##   
## Accuracy : 0.4897   
## 95% CI : (0.4769, 0.5026)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3331   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9104 0.33011 0.49220 0.0000 0.44085  
## Specificity 0.6348 0.91888 0.77856 1.0000 0.99917  
## Pos Pred Value 0.4977 0.49409 0.31942 NaN 0.99168  
## Neg Pred Value 0.9469 0.85109 0.87895 0.8362 0.88805  
## Prevalence 0.2845 0.19354 0.17434 0.1638 0.18386  
## Detection Rate 0.2590 0.06389 0.08581 0.0000 0.08105  
## Detection Prevalence 0.5203 0.12931 0.26865 0.0000 0.08173  
## Balanced Accuracy 0.7726 0.62450 0.63538 0.5000 0.72001

#### Calculate predictions at the validation set

#### Transforming the input to match the input for the models constructed

validation<-read.csv("pml-testing.csv")  
usedfeatures3<-usedfeatures[1:53]  
newvalidation<-validation[usedfeatures3]  
newvalidation<-predict(preobj,newvalidation)   
predictionsval<-predict(modelFit,newdata=newvalidation)  
predictions2val<-predict(modelFit2,newdata=newvalidation)

##### Random forest was the most accurate model at the testing set

##### therefore this model will be used to estimate the values for the validation set

##### The out of sample accuracy would be close to the one calculated

##### of the testing set for the random forest model and the classification results

##### as follows

predictions2val

## [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