Practical Machine Learning

Ryan Jolicoeur

May 15, 2016

# Practical Machine Learning Course Project

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

First we need to load the packages that we will be utilize to complete our analysis and predict the exercises

library(caret);

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart);  
library(rpart.plot);  
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

# Data Preparation

Next, we need to set seed for reproducibility and load in the data

set.seed(976)  
  
train\_data <- read.csv( "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",   
 na.strings = c("NA", "#DIV/0!", ""))  
  
test\_data <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",  
 na.strings = c("NA", "#DIV/0!", ""))

# Data Preparation and Cleansing

We need to split our data into our training and test sets. Given the size I choose a 65% training - 35% test set

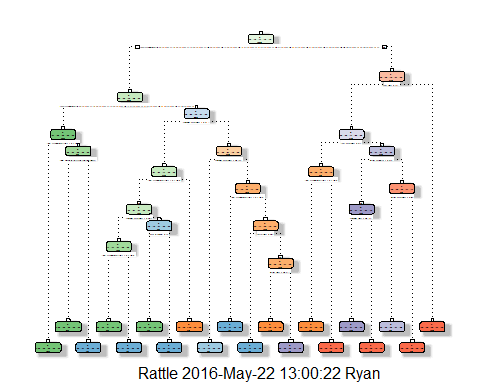
inTrain <-createDataPartition(train\_data$classe, p=0.65, list=FALSE)  
  
training2 <- train\_data[inTrain,]  
testing2 <- train\_data[-inTrain,]  
  
  
nzv <- nearZeroVar(training2, saveMetrics = TRUE )  
training2 <- training2[,nzv$nzv==FALSE]  
  
nzv <- nearZeroVar(testing2,saveMetrics=TRUE)  
testing2 <- testing2[,nzv$nzv==FALSE]  
  
training2 <- training2[c(-1)]  
  
training3 <- training2  
for(i in 1:length(training2)){  
 if(sum (is.na(training2[,i]))/nrow(training2)>=0.65){  
 for(j in 1:length(training3)){  
 if(length(grep(names(training2[i]),names(training3)[j])==1)){  
 training3 <- training3[,-j]  
 }  
 }  
 }  
}  
  
training2 <- training3  
rm(training3)  
data <- colnames(training2)  
data2 <- colnames(training2[,-58])  
testing2 <- testing2[data]  
test\_data <- test\_data[data2]  
  
for (i in 1:length(testing2)){  
 for(j in 1:length(training2)){  
 if( length( grep(names(training2[i]), names(testing2)[j]) ) ==1) {  
 class(testing2[j])<- class(training2[i])  
 }  
 }  
}  
test\_data <- rbind(training2[2,-58],test\_data)  
test\_data<- test\_data[-1,]

# Prediction Models

### Decision Trees

For the first model we will use Decision Trees to predict

set.seed(976)  
DT\_fit <- rpart(classe~., data=training2, method="class")  
fancyRpartPlot(DT\_fit)



DT\_predict <- predict(DT\_fit, testing2, type="class")  
Confusion\_DT <- confusionMatrix(DT\_predict, testing2$classe)  
Confusion\_DT

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1889 53 8 1 0  
## B 60 1215 132 31 0  
## C 4 51 1035 154 5  
## D 0 9 22 878 176  
## E 0 0 0 61 1081  
##   
## Overall Statistics  
##   
## Accuracy : 0.8883   
## 95% CI : (0.8806, 0.8956)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8587   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9672 0.9149 0.8647 0.7804 0.8566  
## Specificity 0.9874 0.9597 0.9622 0.9639 0.9891  
## Pos Pred Value 0.9682 0.8449 0.8287 0.8092 0.9466  
## Neg Pred Value 0.9870 0.9792 0.9712 0.9573 0.9684  
## Prevalence 0.2845 0.1934 0.1744 0.1639 0.1838  
## Detection Rate 0.2752 0.1770 0.1508 0.1279 0.1575  
## Detection Prevalence 0.2842 0.2095 0.1819 0.1580 0.1664  
## Balanced Accuracy 0.9773 0.9373 0.9135 0.8722 0.9228

### Random Forest

For the second model we will use Random Forests to predict

set.seed(976)  
RF\_predict <- randomForest(classe ~ ., data=training2)  
RF\_predict2 <- predict(RF\_predict, testing2, type = "class")  
confusionMatrix(RF\_predict2, testing2$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1953 1 0 0 0  
## B 0 1327 0 0 0  
## C 0 0 1192 1 0  
## D 0 0 5 1124 1  
## E 0 0 0 0 1261  
##   
## Overall Statistics  
##   
## Accuracy : 0.9988   
## 95% CI : (0.9977, 0.9995)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9985   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9992 0.9958 0.9991 0.9992  
## Specificity 0.9998 1.0000 0.9998 0.9990 1.0000  
## Pos Pred Value 0.9995 1.0000 0.9992 0.9947 1.0000  
## Neg Pred Value 1.0000 0.9998 0.9991 0.9998 0.9998  
## Prevalence 0.2845 0.1934 0.1744 0.1639 0.1838  
## Detection Rate 0.2845 0.1933 0.1736 0.1637 0.1837  
## Detection Prevalence 0.2846 0.1933 0.1738 0.1646 0.1837  
## Balanced Accuracy 0.9999 0.9996 0.9978 0.9990 0.9996

# Summary

We can see here as expected the Random Forests provides the most accurate prediction between the two models that we compared.

Hence we will utilize the random forests to make our predictions:

final\_prediction <- predict(RF\_predict, test\_data, type="class")  
final\_prediction

## 1 2 31 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

# Out of sample error

Using our results we get the out of sample error as being: \* 1 - 0.9988 = 0.0012