Practical Machine Learning Project-write up

Geeta Nain

September 25, 2015

## Introduction

* We have training and testing datasets from accelerometers on the belt, forearm, arm, and dumbell of six participants, who participated in dumbell lifting exercise in five different way.The five ways, as described in the study, were Class A (exactly according to the specification), Class B (throwing the elbows to the front) , Class C (lifting the dumbbell only halfway) , class D (lowering the dumbbell only halfway) and Class E (throwing the hips to the front). -Training data consists of accelerometer data (having all these classes) and a label(classe) identifying the quality of the activity the participant was doing. Our testing data consists of accelerometer data (all classes) without the identifying label. so, This report consist mainly following points to predict the manner (lebel) for 20 testing observation.

1.Data preprocessing 2.Building Model using different methods (i.e. rpart, rf etc.) 3.Applying cross validation 4.Estimated out of sample error 4.Predictions

## Data Preprocessing

-Loading packages and importing data

library(AppliedPredictiveModeling)

## Warning: package 'AppliedPredictiveModeling' was built under R version  
## 3.1.3

library(caret)

## Warning: package 'caret' was built under R version 3.1.3

## Loading required package: lattice  
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.1.3

library(rattle)

## Warning: package 'rattle' was built under R version 3.1.3

## Loading required package: RGtk2

## Warning: package 'RGtk2' was built under R version 3.1.3

## Rattle: A free graphical interface for data mining with R.  
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.1.3

## Loading required package: rpart

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.

training <- "pml-training.csv"  
testing <- "pml-testing.csv"  
  
# Importing data considering null values as NA  
training <- read.csv(training, na.strings=c("NA",""), header=TRUE)  
column\_training <- colnames(training)  
  
testing <- read.csv(testing, na.strings=c("NA",""), header=TRUE)  
column\_testing <- colnames(testing)  
  
# Verify that the column names (excluding classe and problem\_id) are identical in the training and test set.  
all.equal(column\_training[1:length(column\_training)-1], column\_testing[1:length(column\_training)-1])

## [1] TRUE

* Partioning the data

Train <- createDataPartition(y=training$classe, p=0.6, list=FALSE)  
trainpart <- training[Train, ];  
testpart <- training[-Train, ]  
dim(trainpart)

## [1] 11776 160

dim(testpart)

## [1] 7846 160

-Removing columns having Nas values , zero variance and which do not make sense for predictions.

# remove variables that are almost always NA  
NAs <- sapply(trainpart, function(x) mean(is.na(x))) > 0.95  
trainpart <- trainpart[, NAs==F]  
testpart <- testpart[, NAs==F]  
  
# remove variables with nearly zero variance  
nzv <- nearZeroVar(trainpart)  
trainpart <- trainpart[, -nzv]  
testpart <- testpart[, -nzv]  
  
# remove variables that don't make sense for prediction (X, user\_name, raw\_timestamp\_part\_1, raw\_timestamp\_part\_2, cvtd\_timestamp, new\_windo, num\_window), which are the first seven variables  
trainpart <- trainpart[, -(1:7)]  
testpart <- testpart[, -(1:7)]

## Building model using different method

### Model1 using method rpart

1. With no extra features.

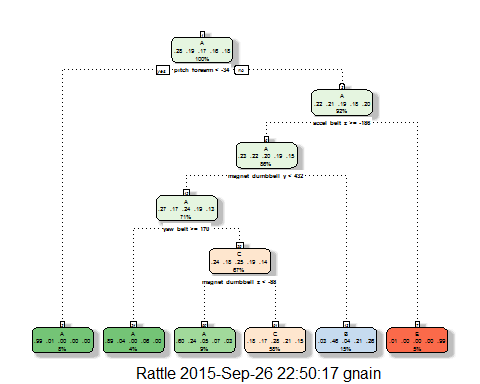
set.seed(666)  
modFit <- train(trainpart$classe ~ ., data =trainpart, method="rpart")  
  
print(modFit, digits=3)

## CART   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...   
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa Accuracy SD Kappa SD  
## 0.0404 0.425 0.240 0.0856 0.145   
## 0.0429 0.403 0.204 0.0842 0.145   
## 0.0542 0.387 0.172 0.0809 0.137   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.0404.

print(modFit$finalModel, digits=3)

## n= 11776   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 11776 8430 A (0.28 0.19 0.17 0.16 0.18)   
## 2) pitch\_forearm< -33.7 987 10 A (0.99 0.01 0 0 0) \*  
## 3) pitch\_forearm>=-33.7 10789 8420 A (0.22 0.21 0.19 0.18 0.2)   
## 6) accel\_belt\_z>=-186 10151 7790 A (0.23 0.22 0.2 0.19 0.15)   
## 12) magnet\_dumbbell\_y< 432 8412 6110 A (0.27 0.17 0.24 0.19 0.13)   
## 24) yaw\_belt>=170 476 50 A (0.89 0.044 0 0.057 0.0042) \*  
## 25) yaw\_belt< 170 7936 5950 C (0.24 0.18 0.25 0.19 0.14)   
## 50) magnet\_dumbbell\_z< -88.5 1106 440 A (0.6 0.24 0.049 0.071 0.034) \*  
## 51) magnet\_dumbbell\_z>=-88.5 6830 4900 C (0.18 0.17 0.28 0.21 0.15) \*  
## 13) magnet\_dumbbell\_y>=432 1739 931 B (0.035 0.46 0.038 0.21 0.26) \*  
## 7) accel\_belt\_z< -186 638 8 E (0.013 0 0 0 0.99) \*

fancyRpartPlot(modFit$finalModel)



predictions <- predict(modFit, newdata=testpart)  
print(confusionMatrix(predictions, testpart$classe), digits=4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1370 228 44 75 19  
## B 47 512 49 239 289  
## C 807 777 1275 972 724  
## D 0 0 0 0 0  
## E 8 1 0 0 410  
##   
## Overall Statistics  
##   
## Accuracy : 0.4546   
## 95% CI : (0.4436, 0.4657)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3166   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6138 0.33729 0.9320 0.0000 0.28433  
## Specificity 0.9348 0.90139 0.4937 1.0000 0.99859  
## Pos Pred Value 0.7892 0.45070 0.2799 NaN 0.97852  
## Neg Pred Value 0.8589 0.85007 0.9717 0.8361 0.86105  
## Prevalence 0.2845 0.19347 0.1744 0.1639 0.18379  
## Detection Rate 0.1746 0.06526 0.1625 0.0000 0.05226  
## Detection Prevalence 0.2213 0.14479 0.5806 0.0000 0.05340  
## Balanced Accuracy 0.7743 0.61934 0.7128 0.5000 0.64146

* Here, there is very low accuracy as (.4827), so I will try including standardization preprocessing\*

1. With only preprocessing.

set.seed(666)  
modFit <- train(trainpart$classe ~ ., preProcess=c("center", "scale"), data = trainpart, method="rpart")  
print(modFit, digits=3)

## CART   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## Pre-processing: centered, scaled   
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...   
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa Accuracy SD Kappa SD  
## 0.0404 0.425 0.240 0.0856 0.145   
## 0.0429 0.403 0.204 0.0842 0.145   
## 0.0542 0.387 0.172 0.0809 0.137   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.0404.

1. With only cross validation.

set.seed(666)  
modFit <- train(trainpart$classe ~ ., trControl=trainControl(method = "cv", number = 4), data = trainpart, method="rpart")  
print(modFit, digits=3)

## CART   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (4 fold)   
##   
## Summary of sample sizes: 8831, 8833, 8831, 8833   
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa Accuracy SD Kappa SD  
## 0.0404 0.477 0.335 0.0199 0.0376   
## 0.0429 0.465 0.319 0.0122 0.0236   
## 0.0542 0.355 0.116 0.0851 0.1412   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.0404.

1. With both preprocessing and cross validation.

set.seed(666)  
modFit <- train(trainpart$classe ~ ., preProcess=c("center", "scale"), trControl=trainControl(method = "cv", number = 4), data =trainpart, method="rpart")  
print(modFit, digits=3)

## CART   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## Pre-processing: centered, scaled   
## Resampling: Cross-Validated (4 fold)   
##   
## Summary of sample sizes: 8831, 8833, 8831, 8833   
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa Accuracy SD Kappa SD  
## 0.0404 0.477 0.335 0.0199 0.0376   
## 0.0429 0.465 0.319 0.0122 0.0236   
## 0.0542 0.355 0.116 0.0851 0.1412   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.0404.

1. Predictions on test part with (4th model) both preprocessing and cross validation.

predictions <- predict(modFit, newdata=testpart)  
print(confusionMatrix(predictions, testpart$classe), digits=4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1370 228 44 75 19  
## B 47 512 49 239 289  
## C 807 777 1275 972 724  
## D 0 0 0 0 0  
## E 8 1 0 0 410  
##   
## Overall Statistics  
##   
## Accuracy : 0.4546   
## 95% CI : (0.4436, 0.4657)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3166   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6138 0.33729 0.9320 0.0000 0.28433  
## Specificity 0.9348 0.90139 0.4937 1.0000 0.99859  
## Pos Pred Value 0.7892 0.45070 0.2799 NaN 0.97852  
## Neg Pred Value 0.8589 0.85007 0.9717 0.8361 0.86105  
## Prevalence 0.2845 0.19347 0.1744 0.1639 0.18379  
## Detection Rate 0.1746 0.06526 0.1625 0.0000 0.05226  
## Detection Prevalence 0.2213 0.14479 0.5806 0.0000 0.05340  
## Balanced Accuracy 0.7743 0.61934 0.7128 0.5000 0.64146

**there is no impact of incorporating both preprocessing and cross validation in accuracy (.4827). lets try using different method for building model**

### Model 2 using method random forest

1. With cross validation

# Train on trainpart with only cross validation.  
set.seed(666)  
modFit <- train(trainpart$classe ~ ., method="rf", trControl=trainControl(method = "cv", number = 4), data=trainpart)  
print(modFit, digits=3)

## Random Forest   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (4 fold)   
##   
## Summary of sample sizes: 8831, 8833, 8831, 8833   
##   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa Accuracy SD Kappa SD  
## 2 0.986 0.983 0.000426 0.000539  
## 26 0.987 0.984 0.002006 0.002538  
## 51 0.983 0.978 0.002458 0.003108  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 26.

1. Prediction using this model against testset.

predictions <- predict(modFit, newdata=testpart)  
print(confusionMatrix(predictions, testpart$classe), digits=4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2226 9 0 0 0  
## B 5 1502 15 0 0  
## C 0 6 1347 16 5  
## D 0 0 6 1269 2  
## E 1 1 0 1 1435  
##   
## Overall Statistics  
##   
## Accuracy : 0.9915   
## 95% CI : (0.9892, 0.9934)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9892   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9973 0.9895 0.9846 0.9868 0.9951  
## Specificity 0.9984 0.9968 0.9958 0.9988 0.9995  
## Pos Pred Value 0.9960 0.9869 0.9803 0.9937 0.9979  
## Neg Pred Value 0.9989 0.9975 0.9968 0.9974 0.9989  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2837 0.1914 0.1717 0.1617 0.1829  
## Detection Prevalence 0.2849 0.1940 0.1751 0.1628 0.1833  
## Balanced Accuracy 0.9979 0.9931 0.9902 0.9928 0.9973

## Predictions 2 based on model 2 (using random forcast method and cross validation)

# predictions based on model build in previous step against 20 testing set provided .  
print(predict(modFit, newdata=testing))

## [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

3.With only both preprocessing and cross validation.

set.seed(666)  
modFit <- train(trainpart$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=trainControl(method = "cv", number = 4), data=trainpart)  
print(modFit, digits=3)

## Random Forest   
##   
## 11776 samples  
## 51 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## Pre-processing: centered, scaled   
## Resampling: Cross-Validated (4 fold)   
##   
## Summary of sample sizes: 8831, 8833, 8831, 8833   
##   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa Accuracy SD Kappa SD  
## 2 0.986 0.982 0.000751 0.000953  
## 26 0.987 0.984 0.001591 0.002013  
## 51 0.982 0.978 0.002591 0.003275  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 26.

1. Prediction using this model against testpart

predictions <- predict(modFit, newdata=testpart)  
print(confusionMatrix(predictions, testpart$classe), digits=4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2227 11 0 0 0  
## B 4 1500 13 0 0  
## C 0 6 1349 15 5  
## D 0 0 6 1271 3  
## E 1 1 0 0 1434  
##   
## Overall Statistics  
##   
## Accuracy : 0.9917   
## 95% CI : (0.9895, 0.9936)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9895   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9978 0.9881 0.9861 0.9883 0.9945  
## Specificity 0.9980 0.9973 0.9960 0.9986 0.9997  
## Pos Pred Value 0.9951 0.9888 0.9811 0.9930 0.9986  
## Neg Pred Value 0.9991 0.9972 0.9971 0.9977 0.9988  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2838 0.1912 0.1719 0.1620 0.1828  
## Detection Prevalence 0.2852 0.1933 0.1752 0.1631 0.1830  
## Balanced Accuracy 0.9979 0.9927 0.9910 0.9935 0.9971

### Predictions against 20 testing set observation based on final model

print(predict(modFit, newdata=testing))

## [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

*Preprocessing actually rose the accuracy rate from 0.9904 to 0.9908 against the training set. Thus I decided to apply both preprocessing and cross validation to final model.*

### Out of sample error

* Random Forest (preprocessing and cross validation) Testpart (part from training data set) : 1-.9908=0.012

### Predictions

* B A B A A E D B A A B C B A E E A B B B