Practical Machine Learning Course Project

Ahmed

April 10, 2016

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement  a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

Data

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>

The data for this project come from this source: <http://groupware.les.inf.puc-rio.br/har>. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

# Solution

### Load the needed libraries

we will load the needed libraries for the project

library (rpart)  
library(ggplot2)  
library (caret)

## Loading required package: lattice

library(rpart.plot)

### Download & Load the data training and testing

download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "Data/pml-training.csv")  
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "Data/pml-testing.csv")  
pml\_train <- read.csv('Data/pml-training.csv', na.strings=c("NA","#DIV/0!", ""))  
pml\_test <- read.csv('Data/pml-testing.csv' , na.strings=c("NA","#DIV/0!", ""))

### Quick Exploration

dim(pml\_train)

## [1] 19622 160

dim(pml\_test)

## [1] 20 160

### Data Munging ( cleansing and preparation)

remove features that have only one unique value or have few unique values relative to the number of samples using nearZeroVar function. Also,remove columns that doesn't have any effect on output. Remove columns with missing values.

bad\_Columns <- nearZeroVar(pml\_train, saveMetrics = TRUE)  
pml\_train <- pml\_train[, bad\_Columns$nzv==FALSE]  
pml\_test <- pml\_test[, bad\_Columns$nzv==FALSE]  
dim(pml\_train)

## [1] 19622 124

unnecessary\_cols <- c("user\_name", "raw\_timestamp\_part\_1","raw\_timestamp\_part\_2", "cvtd\_timestamp")  
for (col in unnecessary\_cols) {  
 pml\_train[, col] <- NULL  
 pml\_test[, col] <- NULL  
}  
dim(pml\_train)

## [1] 19622 120

pml\_train<-pml\_train[,colSums(is.na(pml\_train)) == 0]  
pml\_test <-pml\_test[,colSums(is.na(pml\_test)) == 0]  
pml\_train$X <- NULL  
pml\_test$X <- NULL  
dim(pml\_train)

## [1] 19622 54

### Prepare data for model

Partition training data into 2 partitions , 75% training and 25% validation

set.seed(234334)  
inTrain <- createDataPartition(y=pml\_train$classe, p=0.75, list=FALSE)  
training <- pml\_train[inTrain, ]   
validation <- pml\_train[-inTrain, ]  
dim(training)

## [1] 14718 54

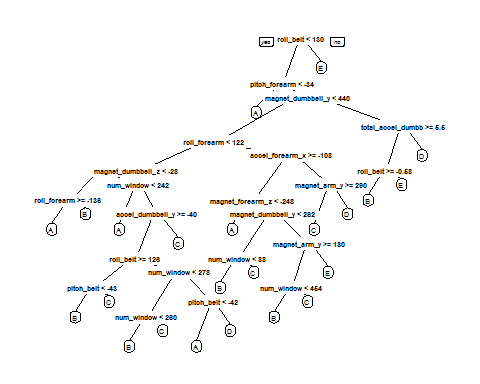
dim(validation)

## [1] 4904 54

### Create the first model

Use Decision trees to build the first model,and test the result against the validation set

modelfit1 <- rpart(classe ~ ., data=training, method="class")  
valid\_predict1 <- predict(modelfit1, validation, type = "class")  
  
rpart.plot(modelfit1)



confusionMatrix(valid\_predict1, validation$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1243 181 33 52 45  
## B 49 548 34 73 75  
## C 14 76 707 67 56  
## D 85 111 51 565 123  
## E 4 33 30 47 602  
##   
## Overall Statistics  
##   
## Accuracy : 0.7473   
## 95% CI : (0.7349, 0.7595)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6795   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8910 0.5774 0.8269 0.7027 0.6681  
## Specificity 0.9114 0.9416 0.9474 0.9098 0.9715  
## Pos Pred Value 0.7999 0.7035 0.7685 0.6043 0.8408  
## Neg Pred Value 0.9546 0.9028 0.9629 0.9398 0.9286  
## Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Detection Rate 0.2535 0.1117 0.1442 0.1152 0.1228  
## Detection Prevalence 0.3169 0.1588 0.1876 0.1907 0.1460  
## Balanced Accuracy 0.9012 0.7595 0.8871 0.8062 0.8198

### Create the second model

Use random forest to build the second model, and test the result against the validation set

modelcontrol <- trainControl(method="cv", number=3, verboseIter=FALSE)  
modelfit2 <- train(classe ~ ., data=training, method="rf",  
 trControl=modelcontrol)

## Loading required package: 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

modelfit2$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 27  
##   
## OOB estimate of error rate: 0.17%  
## Confusion matrix:  
## A B C D E class.error  
## A 4184 0 0 0 1 0.0002389486  
## B 4 2840 4 0 0 0.0028089888  
## C 0 3 2561 3 0 0.0023373588  
## D 0 0 5 2406 1 0.0024875622  
## E 0 1 0 3 2702 0.0014781966

valid\_predict2 <- predict(modelfit2, newdata=validation)  
confusionMatrix(valid\_predict2, validation$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1394 1 0 0 0  
## B 1 948 1 0 0  
## C 0 0 854 2 0  
## D 0 0 0 802 3  
## E 0 0 0 0 898  
##   
## Overall Statistics  
##   
## Accuracy : 0.9984   
## 95% CI : (0.9968, 0.9993)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9979   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9993 0.9989 0.9988 0.9975 0.9967  
## Specificity 0.9997 0.9995 0.9995 0.9993 1.0000  
## Pos Pred Value 0.9993 0.9979 0.9977 0.9963 1.0000  
## Neg Pred Value 0.9997 0.9997 0.9998 0.9995 0.9993  
## Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Detection Rate 0.2843 0.1933 0.1741 0.1635 0.1831  
## Detection Prevalence 0.2845 0.1937 0.1746 0.1642 0.1831  
## Balanced Accuracy 0.9995 0.9992 0.9992 0.9984 0.9983

### Apply the agorithm on the testing set 20 records to predict the outcome level

predicttest <- predict(modelfit2, newdata=pml\_test)  
predicttest

## [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

## Conclusion

Decision

Random Forest algorithm performed better than Decision Trees. Prediction evaluations were based on maximizing the accuracy and minimizing the out-of-sample error. after cleaning the data and removing the unrelated, unrelated columns, both models were tested and found the model with best accuracy and apply it on the testing data.

## References

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6\_6. Cited by 2 (Google Scholar)

Read more: <http://groupware.les.inf.puc-rio.br/har#ixzz45UOuBobE>