Machine\_learning

CW

15/04/2018

In this project, we aim to use data from accelerometers of 6 participants performing barbell lifts correctly and incorrectly in 5 different ways to predict how well they do it. outcome variable: classe \* 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 \* Class E - throwing the hips to the front

setwd("/Volumes/Daisy/R/R\_assignmant/machine\_learning/")  
library(caret)  
library(randomForest)  
library(rpart)  
library(rpart.plot)

trainingV <- read.csv("pml-training.csv",head=T)  
testingo <- read.csv("pml-testing.csv",head=T)  
set.seed(1121)  
table(is.na(trainingV))

##   
## FALSE TRUE   
## 1852048 1287472

There many a lot of missing values, thus we first exclude variables with limited variance in prediction in training set. Same variables will be used in the testing set.

# Preprocessing

1. exclude variables with many missing values. The “No.missingVar” variable will return number of missing values for all variables.

sum(is.na(trainingV))

## [1] 1287472

missingN<-matrix()  
  
matName<-names(trainingV)  
for (i in 1:dim(trainingV)[2]) {  
 t<-sum(is.na(trainingV[,i]))  
 missingN <- c(missingN,t)  
}  
  
missingN <- missingN[!is.na(missingN)]  
No.missingVar <- data.frame(missingN, matName)  
validVar<-No.missingVar[No.missingVar$missingN ==0,]  
trainingV2 <- trainingV[names(trainingV) %in% validVar$matName]  
trainingV3 <- trainingV2[,-c(1,2)]

1. The next step is to exclude variables with zero variance (little variance)

varT<-nearZeroVar(trainingV3,saveMetrics = T)  
trainingV4<- trainingV3[,varT$nzv == FALSE]

1. Subsequently, the training dataset was seperated to a sub-training set and a validation set.

valida\_index <- createDataPartition(trainingV4$classe, p =.75, list = FALSE)  
validation <- trainingV4[-valida\_index,]  
training <- trainingV4[valida\_index,]  
testing <- testingo[,names(testingo) %in% names(training)]  
  
dim(validation)

## [1] 4904 57

dim(training)

## [1] 14718 57

dim(testing)

## [1] 20 56

# Creating models.

1. Here I tested three models, random forest (“rf”), rpart and gbm.

modf1 <- train(classe ~., training, method ="rf")  
modf2 <- train (classe ~., training, method ="rpart")  
modf3 <- train (classe ~., training, method ="gbm")

1. This is to predict test the model using the models created in the previous step on the validation set.

crossV1<- predict(modf1,newdata = validation)  
crossV2<- predict(modf2,newdata = validation)  
crossV3<- predict(modf3,newdata = validation)

1. To measure the model fit, using validation set:

confusionMatrix(crossV1,validation$classe)$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 1.0000000 1.0000000 0.9992481 1.0000000 0.2844617   
## AccuracyPValue McnemarPValue   
## 0.0000000 NaN

confusionMatrix(crossV2,validation$classe)$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 5.010196e-01 3.744276e-01 4.869265e-01 5.151114e-01 2.844617e-01   
## AccuracyPValue McnemarPValue   
## 2.213235e-223 NaN

confusionMatrix(crossV3,validation$classe)$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.9973491 0.9966468 0.9954712 0.9985878 0.2844617   
## AccuracyPValue McnemarPValue   
## 0.0000000 NaN

Confusion Matrix shows that model 1 (random forest, method = “rf”) and model 3 (medho = “rpart”) accurately predict the validation dataset. We thus will apply the two models to the testing set.

# predicting the testing set

predict(modf1,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

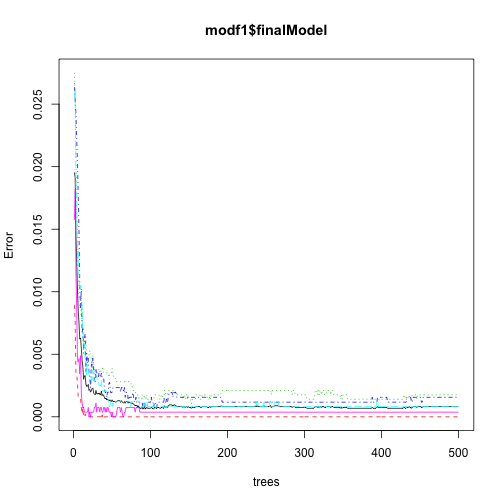
predict(modf3,newdata=testing)

## [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

The predictions made by the two models are identical

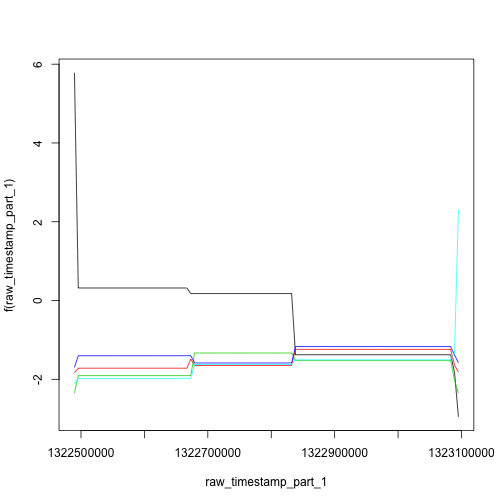
# Appendix

plot(modf1$finalModel)



plot of chunk unnamed-chunk-10

plot(modf3$finalModel)



plot of chunk unnamed-chunk-10