Analysis of Accelerometers Data

Sarah

9/17/2020

Introduction

This project uses data recorded from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. These participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal is to predict the manner in which the participants did the exercise » classe variable

Getting Data

The training and testing sets are downloaded from the online source as below:

```
set.seed(111)
```

```
if(!file.exists("data")){
        dir.create("`data")
        download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",destfil
        download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",destfile
}
training<-read.csv("data/training.csv")
testing<-read.csv("data/testing.csv")
```

Exploring the training set

In order to build the model we should first explore the training set. The training set consists of 160 columns and 19622 rows.

```
checkNAs<-function(v,len){
        s1<-sum(is.na(v)*1)/len
        s2<-0
        if(class(v)=="factor"){
          s2<-sum((v=="")*1)/len
        }
        max(s1,s2)
}
amountNAs<-sapply(training,FUN = checkNAs,nRows)
colsNAs<-amountNAs[which(amountNAs>0)]
```

1- By looking at the data, we first noticed 100 columns with high percentage of missing values so we decided to track them and exclude them from the data set:

```r
colsNAs
```

```
##       kurtosis_roll_belt      kurtosis_picth_belt        kurtosis_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##       skewness_roll_belt     skewness_roll_belt.1        skewness_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##            max_roll_belt           max_picth_belt             max_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##            min_roll_belt           min_pitch_belt             min_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##      amplitude_roll_belt     amplitude_pitch_belt       amplitude_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##     var_total_accel_belt            avg_roll_belt         stddev_roll_belt 
##                0.9793089                0.9793089                0.9793089 
##            var_roll_belt           avg_pitch_belt        stddev_pitch_belt 
##                0.9793089                0.9793089                0.9793089 
##           var_pitch_belt             avg_yaw_belt          stddev_yaw_belt 
##                0.9793089                0.9793089                0.9793089 
##             var_yaw_belt            var_accel_arm             avg_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##          stddev_roll_arm             var_roll_arm            avg_pitch_arm 
##                0.9793089                0.9793089                0.9793089 
##         stddev_pitch_arm            var_pitch_arm              avg_yaw_arm 
##                0.9793089                0.9793089                0.9793089 
##           stddev_yaw_arm              var_yaw_arm        kurtosis_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##       kurtosis_picth_arm        kurtosis_yaw_arm        skewness_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##       skewness_pitch_arm        skewness_yaw_arm             max_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##            max_picth_arm              max_yaw_arm             min_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##            min_pitch_arm              min_yaw_arm       amplitude_roll_arm 
##                0.9793089                0.9793089                0.9793089 
##      amplitude_pitch_arm        amplitude_yaw_arm   kurtosis_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##  kurtosis_picth_dumbbell    kurtosis_yaw_dumbbell   skewness_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##  skewness_pitch_dumbbell    skewness_yaw_dumbbell        max_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##       max_picth_dumbbell        max_yaw_dumbbell        min_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##       min_pitch_dumbbell        min_yaw_dumbbell  amplitude_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
## amplitude_pitch_dumbbell   amplitude_yaw_dumbbell       var_accel_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##        avg_roll_dumbbell     stddev_roll_dumbbell        var_roll_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##       avg_pitch_dumbbell    stddev_pitch_dumbbell       var_pitch_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##         avg_yaw_dumbbell      stddev_yaw_dumbbell         var_yaw_dumbbell 
##                0.9793089                0.9793089                0.9793089 
##    kurtosis_roll_forearm   kurtosis_picth_forearm     kurtosis_yaw_forearm 
##                0.9793089                0.9793089                0.9793089 
```

```
##    skewness_roll_forearm   skewness_pitch_forearm     skewness_yaw_forearm
##                0.9793089                0.9793089                0.9793089
##         max_roll_forearm        max_picth_forearm          max_yaw_forearm
##                0.9793089                0.9793089                0.9793089
##         min_roll_forearm        min_pitch_forearm          min_yaw_forearm
##                0.9793089                0.9793089                0.9793089
##   amplitude_roll_forearm  amplitude_pitch_forearm    amplitude_yaw_forearm
##                0.9793089                0.9793089                0.9793089
##        var_accel_forearm         avg_roll_forearm      stddev_roll_forearm
##                0.9793089                0.9793089                0.9793089
##         var_roll_forearm        avg_pitch_forearm     stddev_pitch_forearm
##                0.9793089                0.9793089                0.9793089
##        var_pitch_forearm          avg_yaw_forearm       stddev_yaw_forearm
##                0.9793089                0.9793089                0.9793089
##          var_yaw_forearm
##                0.9793089
```

```r
trainingNew<-training[,!(names(training) %in% names(colsNAs))]
testingNew<-testing[,!(names(testing) %in% names(colsNAs))]
```

Now we are left with 60 columns
2- we will exclude the first 7 columns as they are related to participants information and other information not usefull to be used as predictors.

```r
excludedCols<-names(trainingNew)[1:7]
excludedCols
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```r
trainingNew<-trainingNew[,!(names(trainingNew) %in% excludedCols)]
testingNew<-testingNew[,!(names(testingNew) %in% excludedCols)]
```

Now we have 53 columns that will try to fit model with

Building the models

Extracting validation set

We split the training set to training and validation. The testing set is left for final testing on unseen data. The validation set is used to evaluate accuracy.

```r
library(caret)
inTrain <- createDataPartition(y = trainingNew$classe, p = 0.8, list = FALSE)
trainingNew <- trainingNew[inTrain, ]
validation <- trainingNew[-inTrain, ]

dim(trainingNew);dim(validation)
```

```
## [1] 15699    53
```

```
## [1] 3143   53
```

Define CV function

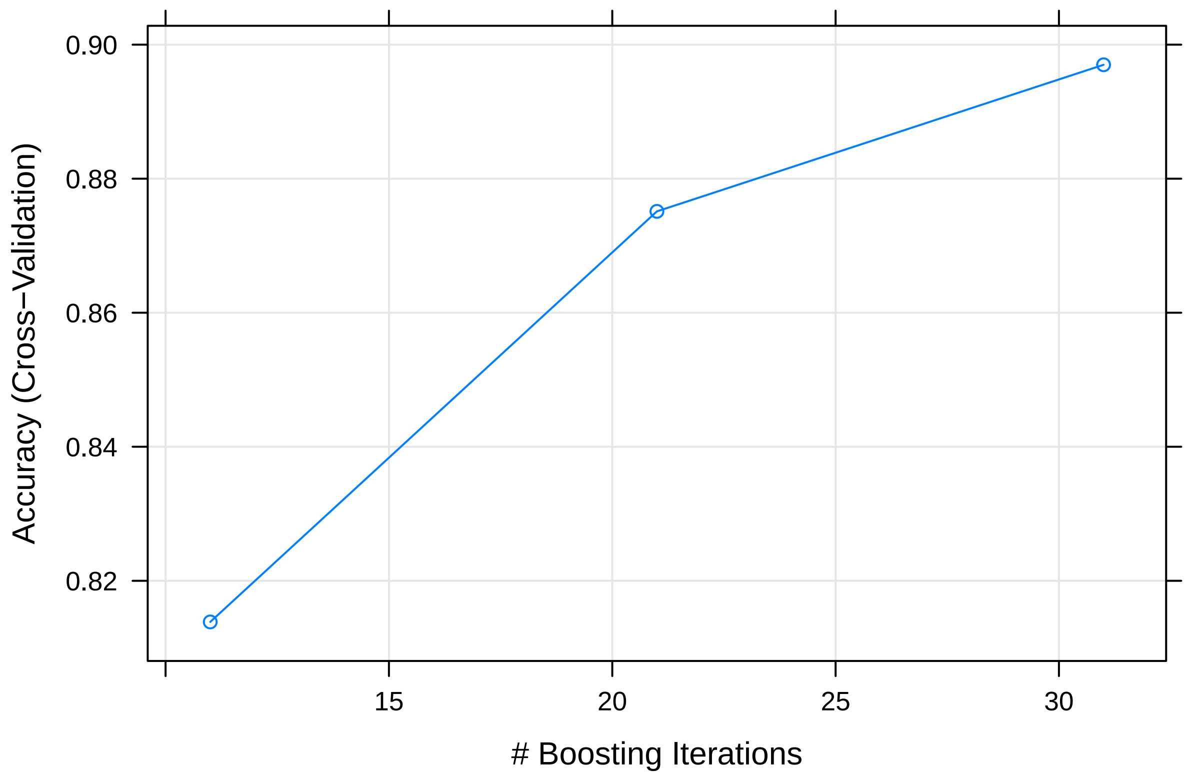
This cross validation function will be used in the train function for all models

```
fitControl <- trainControl(method = "cv", number = 5, returnResamp = "all")
```

Train model :logistic regression with boosting

```
mod.LogitBoost<- train(classe ~ ., data = trainingNew, method = "LogitBoost"
              , trControl = fitControl)
```

```
plot(mod.LogitBoost)
```

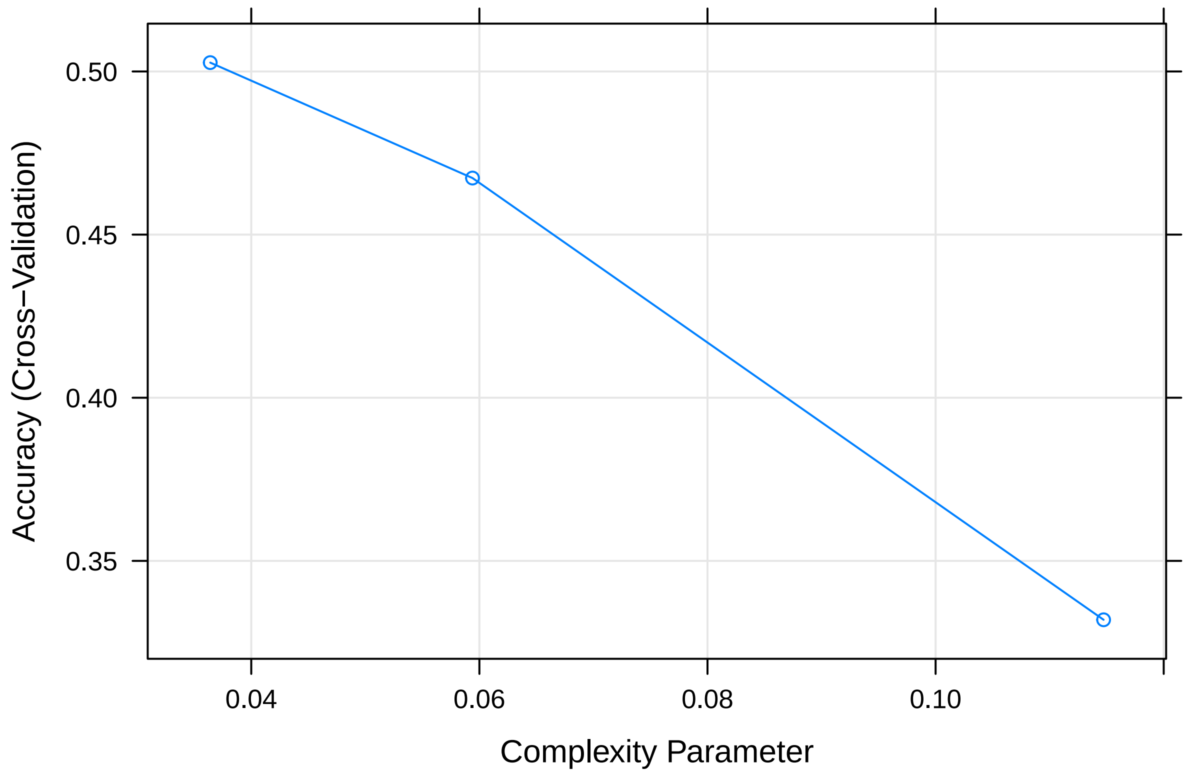


The LogitBoost shows relatively good accuracy over all cross validation sets

Train model :classification tree

```
mod.rpart<- train(classe ~ ., data = trainingNew, method = "rpart",
                  trControl = fitControl)
```

```r
plot(mod.rpart)
```



The rpart model shows worse results than LogitBoost over all the cross validation sets.

Evaluate on validation set

LogitBoost

```r
pred.valid.LogitBoost<-predict(mod.LogitBoost,validation)
confusionMatrix(pred.valid.LogitBoost,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A   B   C   D   E
##            A 812  53   9   7   3
##            B  11 436  25   3   9
##            C   3  33 402  26  12
##            D   5   4  17 386  15
##            E   5   0   5   9 467
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9079
##                  95% CI : (0.8965, 0.9184)
##     No Information Rate : 0.3032
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8825
##
##  Mcnemar's Test P-Value : 6.451e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9713   0.8289   0.8777   0.8956   0.9229
## Specificity            0.9625   0.9785   0.9678   0.9824   0.9916
## Pos Pred Value         0.9186   0.9008   0.8445   0.9040   0.9609
## Neg Pred Value         0.9872   0.9604   0.9754   0.9807   0.9828
## Prevalence             0.3032   0.1908   0.1661   0.1563   0.1835
## Detection Rate         0.2945   0.1581   0.1458   0.1400   0.1694
## Detection Prevalence   0.3206   0.1756   0.1727   0.1549   0.1763
## Balanced Accuracy      0.9669   0.9037   0.9228   0.9390   0.9572
```

classification tree

```
pred.valid.rpart<-predict(mod.rpart,validation)
confusionMatrix(pred.valid.rpart,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   A   B   C   D   E
##          A 821 249 256 234  87
##          B  10 216  22  97  72
##          C  55 162 260 178 148
##          D   0   0   0   0   0
##          E   1   0   0   0 275
##
## Overall Statistics
##
##                Accuracy : 0.5002
##                  95% CI : (0.4825, 0.5178)
##     No Information Rate : 0.2822
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3472
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9256  0.34450  0.48327   0.0000  0.47251
## Specificity            0.6339  0.92011  0.79155   1.0000  0.99961
```

```
## Pos Pred Value         0.4985  0.51799  0.32379     NaN  0.99638
## Neg Pred Value         0.9559  0.84923  0.88120  0.8381  0.89292
## Prevalence             0.2822  0.19949  0.17117  0.1619  0.18517
## Detection Rate         0.2612  0.06872  0.08272  0.0000  0.08750
## Detection Prevalence   0.5240  0.13268  0.25549  0.0000  0.08781
## Balanced Accuracy      0.7797  0.63230  0.63741  0.5000  0.73606
```

Prediction on test set

LogitBoost

```
pred.test.LogitBoost<-predict(mod.LogitBoost,testingNew)
```

The "logistic regression with boosting" model give a very high accuracy (>90%) on validation and testing set with only few classes predicted wrong.

In sample vs. out of sample error of LogitBoost (best accuracy model)

```
#In Sample
pred.train.LogitBoost<-predict(mod.LogitBoost,trainingNew)
NAS<- which(is.na(pred.train.LogitBoost))
1-sum((pred.train.LogitBoost[-NAS]==trainingNew[-NAS,'classe'])*1)/length(pred.train.LogitBoost)
```

```
## [1] 0.212625
```

```
#Out Sample
NAS<- which(is.na(pred.valid.LogitBoost))
1-sum((pred.valid.LogitBoost[-NAS]==validation[-NAS,'classe'])*1)/length(pred.valid.LogitBoost)
```

```
## [1] 0.2036271
```

The out of sample error is higher than the in sample error because it was tested on new unseen data but generaly it is very low