Predicting the manner in which an exercise is executed

Practical Machine Learning course project

by Ruja Babacheva

June 22 2018

Synopsis

My goal in this report is to fit a model to predict the manner in which the barbell lift exercise is done using training data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The variable *classe* contains the manner - class A corresponds to the specified execution, classes B, C, D, E correspond to common mistakes. Then I used the model to predict the 20 test cases in the test data. I used 52 predictors and tried 3 models - decision tree, random forest and generalized boosted. The random forest model shows the best results and I applied it to predict the cases in test data.

Data source

The data for this project come from this source: [http://groupware.les.inf.puc-rio.br/har]. Published: *Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013*

Loading data

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
urlTraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- read.csv(url(urlTraining), na.strings=c("NA",""))
dim(training)
```

```
## [1] 19622   160
```

```
urlTesting <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- read.csv(url(urlTesting), na.strings=c("NA",""))
dim(testing)
```

```
## [1]  20 160
```

Both training and testing data have 160 variables.

Cross validation

For building the model I use training data. I splitted it to train and test partitions.

```
set.seed(111)
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
myTrain <- training[inTrain, ]
myTest  <- training[-inTrain, ]
dim(myTrain)
```

```
## [1] 13737   160
```

```
dim(myTest)
```

```
## [1] 5885  160
```

Cleaning data

First I cleaned myTrain set, then I applied the same cleaning procedure to myTest.

First 7 variables contain information useless for the model - row numbers, user names, time stamps, windows.

```
names(myTrain)[1:7]
```

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

So I removed them.

```
myTrain <- myTrain[, -c(1:7)]
```

Next I removed the variables with too many NA's.

```
myTrain <- myTrain[, !colMeans(is.na(myTrain)>.9)]
dim(myTrain)
```

```
## [1] 13737    53
```

Now in MyTrain left 53 variables. Also I checked for near zero variance variables.

```
nzv <- nearZeroVar(myTrain, saveMetrics=TRUE)
myTrain <- myTrain[, nzv$nzv==FALSE]
dim(myTrain)
```

```
## [1] 13737    53
```

There are not variables with near zero variance, so ultimately I use the set with 53 variables for building the models. I reduced the test partition myTest to the same 53 variables.

```
myTest <- myTest[, names(myTrain)]
```

Building models

I tried 3 models - decision tree, random forest and generalized boosted.

Decision tree

```
modelDT <- train(classe ~., data= myTrain, method="rpart")
predictDT <- predict(modelDT, myTest)
confusionMatrix(predictDT, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1518  478  457  417  171
##          B   24  379   37  180  134
##          C  126  282  532  367  283
##          D    0    0    0    0    0
##          E    6    0    0    0  494
##
## Overall Statistics
##
##                Accuracy : 0.4967
##                  95% CI : (0.4838, 0.5095)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3425
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068   0.3327   0.5185   0.0000  0.45656
## Specificity            0.6383   0.9210   0.7823   1.0000  0.99875
## Pos Pred Value         0.4992   0.5027   0.3346      NaN  0.98800
## Neg Pred Value         0.9451   0.8519   0.8850   0.8362  0.89081
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.2579   0.0644   0.0904   0.0000  0.08394
## Detection Prevalence   0.5167   0.1281   0.2702   0.0000  0.08496
## Balanced Accuracy      0.7726   0.6269   0.6504   0.5000  0.72766
```

This model gives only 50% accuracy.

Random forest

```
modelRF <- train(classe ~., data= myTrain, method="rf", trControl=trainControl(method = "cv",
number = 3))
```

```
predictRF <- predict(modelRF, myTest)
confusionMatrix(predictRF, myTest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    1 1129    3    0    0
##          C    0    9 1022   24    0
##          D    0    0    1  939    4
##          E    0    0    0    1 1078
## 
## Overall Statistics
## 
##                Accuracy : 0.9925
##                  95% CI : (0.99, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9912   0.9961   0.9741   0.9963
## Specificity            0.9998   0.9992   0.9932   0.9990   0.9998
## Pos Pred Value         0.9994   0.9965   0.9687   0.9947   0.9991
## Neg Pred Value         0.9998   0.9979   0.9992   0.9949   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1918   0.1737   0.1596   0.1832
## Detection Prevalence   0.2845   0.1925   0.1793   0.1604   0.1833
## Balanced Accuracy      0.9996   0.9952   0.9947   0.9865   0.9980
```

Random forest model gives more than 99% accuracy.

Generalized boosted model

```
modelGBM <- train(classe ~., data= myTrain, method="gbm", trControl=trainControl(method="repea
tedcv", number=3, repeats=2), verbose=FALSE)
predictGBM <- predict(modelGBM, myTest)
confusionMatrix(predictGBM, myTest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1650   34    0    0    0
##          B   18 1076   34    4   10
##          C    2   29  969   33    6
##          D    3    0   19  923   18
##          E    1    0    4    4 1048
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9628
##                  95% CI : (0.9576, 0.9675)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9529
##  Mcnemar's Test P-Value : 3.235e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9447   0.9444   0.9575   0.9686
## Specificity            0.9919   0.9861   0.9856   0.9919   0.9981
## Pos Pred Value         0.9798   0.9422   0.9326   0.9585   0.9915
## Neg Pred Value         0.9943   0.9867   0.9882   0.9917   0.9930
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2804   0.1828   0.1647   0.1568   0.1781
## Detection Prevalence   0.2862   0.1941   0.1766   0.1636   0.1796
## Balanced Accuracy      0.9888   0.9654   0.9650   0.9747   0.9834
```

The accuracy with this model is 96%, a bit worse than random forest.

Results

Although the random forest model takes more time it gives the best accuricy - almost 100%, so I choose this model to predict the 20 cases in testing data.

```
predictions <- predict(modelRF, newdata=testing)
```

My predictions are: B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B