Human Activity Recognition: Multiclass Classification

cyanophyta

August 12, 2017

Introduction

We are studying Human Activity Recognition data here, collected from this archived web site: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. Data set here contains various measurements gathered from devices attached to bodies of participants while they were performing barbell lifts in five different positions such as Sitting, Sitting down, Standing, Standing up and Walking.

We'll try to fit a number of models with the aim of predicting the varible that indicates the position of the participant while performing the activity. In training set, this position value is known and is marked as 'classe' variable.

Loading data

We load training set from pml-training.csv file (url: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and test data from pml-testing.csv file (url: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv). As NA values are marked here as either empty string or as "NA" or sometimes as "#DIV/0!", we read the csv files noting these possible values of NA strings.

```
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)
library(caret, warn.conflicts = FALSE, quietly = TRUE)
library(rpart, warn.conflicts = FALSE, quietly = TRUE)
library(randomForest, warn.conflicts = FALSE, quietly = TRUE)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
trainset <- read.csv("pml-training.csv", na.strings=c("#DIV/0!", "", "NA"))
testset<- read.csv("pml-testing.csv", na.strings=c("#DIV/0!", "", "NA"))

dim(trainset)
```

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

```
dim(testset)
```

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

Cleaning and transforming data

While exploring variables in training set, we notice that the first seven columns are not required for prediction goal here, as these variables are not measurements which can help in identifying 'classe' variable. Once these variables are removed, we look for presence of NA in measurements and we remove the variables with NA values from our set of predictors.

We do this cleaning exercise on both training and test sets. Finally we note down the dimensions of both training and test sets.

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

nacols <- function(df) {
  colnames(df)[unlist(lapply(df, function(x) anyNA(x)))]
}

nacolList <- nacols(trainset)

subtrainset <- trainset %>%
  select(-one_of(nacolList))

subtestset <- testset %>%
  select(-one_of(nacolList))

dim(subtrainset)
```

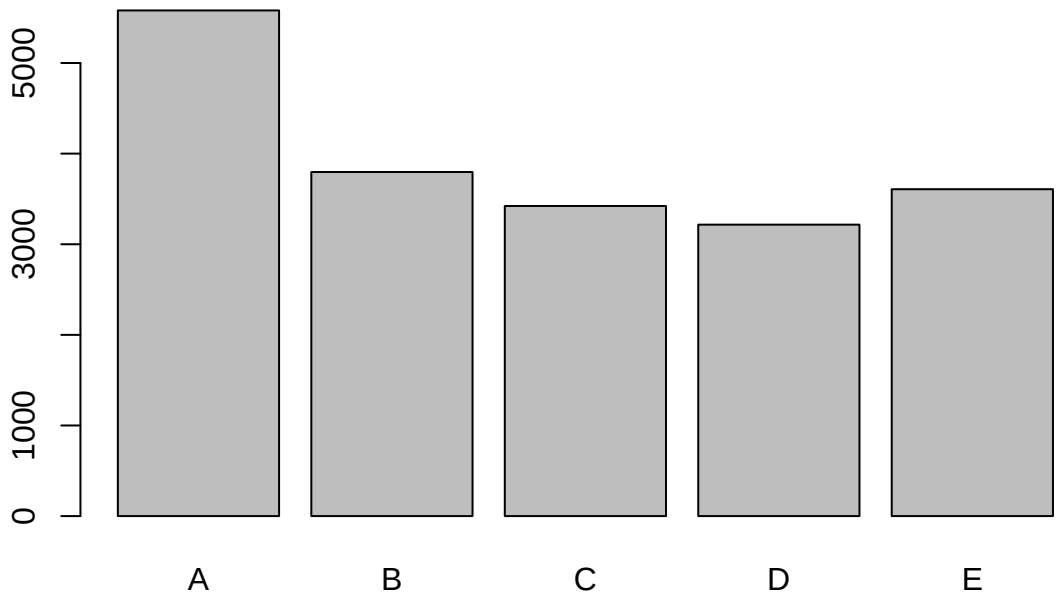
```
## [1] 19622    53
```

```
dim(subtestset)
```

```
## [1] 20 53
```

Cleanup steps reduced the list of predictor candidates to 53, whereas the original sets had 160 variables. Now, we plot a bar chart of 'classe' variable from training set.

```
plot(subtrainset$classe)
```



Data preparation for cross-validation

We divide the training set into two parts, in 3:1 ratio, to form training set named 'training' and cross-validation set named 'testing'. As cross-validation set is not used for building models, accuracy measured for prediction here will measure out-of-sample errors for fitted models.

From this point we set a seed of 2017 for reproducibility purpose.

```
set.seed(2017)
inTrain <- createDataPartition(y=subtrainset$classe, p=0.75, list=FALSE)
training <- subtrainset[inTrain,]
testing <- subtrainset[-inTrain,]

dim(training)
```

```
## [1] 14718    53
```

```
dim(testing)
```

```
## [1] 4904   53
```

Prediction using Recursive Partitioning

Our first attempt to fit a prediction model is to use rpart model from rpart package of R.

```
modTree <- rpart(classe ~ ., data = training, method = "class")
predTree <- predict(modTree, testing, type = "class")
confusionMatrix(predTree, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1215  168   27   45   45
##          B   61  599   81   64   96
##          C   22   53  621  122   81
##          D   83   70   67  523   95
##          E   14   59   59   50  584
##
## Overall Statistics
##
##                Accuracy : 0.7223
##                  95% CI : (0.7095, 0.7348)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6479
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8710   0.6312   0.7263   0.6505   0.6482
## Specificity            0.9188   0.9236   0.9313   0.9232   0.9545
## Pos Pred Value         0.8100   0.6648   0.6908   0.6241   0.7624
## Neg Pred Value         0.9471   0.9126   0.9416   0.9309   0.9234
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2478   0.1221   0.1266   0.1066   0.1191
## Detection Prevalence   0.3059   0.1837   0.1833   0.1709   0.1562
## Balanced Accuracy      0.8949   0.7774   0.8288   0.7868   0.8014
```

As the Confusion Matrix output shows, it has an accuracy of 0.7223 with 95% Confidence interval being (0.7095, 0.7348).

Prediction using Random Forest

Our second attempt is made using Random Forest model from randomForest package of R.

```
modRF <- randomForest(classe ~ ., data = training, method = "class")
predRF <- predict(modRF, testing, type = "class")
confusionMatrix(predRF, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    7    0    0    0
##          B    0  941    2    0    0
##          C    0    1  852    7    2
##          D    0    0    1  797    4
##          E    1    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9916   0.9965   0.9913   0.9933
## Specificity            0.9980   0.9995   0.9975   0.9988   0.9998
## Pos Pred Value         0.9950   0.9979   0.9884   0.9938   0.9989
## Neg Pred Value         0.9997   0.9980   0.9993   0.9983   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1919   0.1737   0.1625   0.1825
## Detection Prevalence   0.2857   0.1923   0.1758   0.1635   0.1827
## Balanced Accuracy      0.9986   0.9955   0.9970   0.9950   0.9965
```

As the Confusion Matrix output shows here, this random forest model produces an accuracy of 0.9949 with 95% Confidence Interval : (0.9925, 0.9967)

Conclusion

First model (rpart) has an accuracy of 0.7223 on cross-validation set. So, it has out-of-sample error (1 - 0.7223 =) 0.2777. Second model (random forest) has an accuracy of 0.9949 on cross-validation set. So, it has an out-of-sample error (1 - 0.9949 =) 0.0051.

As random forest model has much better accuracy and very low out-of-sample error, we chose this model for prediction on testset, as the expected misclassification error is very low.

```
predict(modRF, subtestset, type = "class")
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```