PracticalMachineLearning

Ludovic

May 21, 2019

Executive summary

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise.

This is the "classe" variable in the training set.

All code and detailed results can be found in appendix.

Pre-work: Package and Data

First we are going to load all the package we need:

```
library(caret)
library(ggplot2)
library(dplyr)

library(rpart)
library(randomForest)
```

Now let us look at the data we want to consider. We have a training set and a testing set we are going to look at the training set and construct our model on it and will test our out-of-sample on the testing set.

```
rawdatatrain <- read.csv("training.csv",sep=",",header = TRUE,quote = "", na.strings = c("NA",""," \" \
rawdatatest <- read.csv("testing.csv",sep=",",header = TRUE,quote = "", na.strings = c("NA",""," \" \"
```

Now we are going to process the data. First we are going to remove the na's columns to remove irrelevant variable. Then we are going to also remove the one that have a variance near zero since they will not have a big impact on the model. Finally by looking at the remaining variable we see that the first column are also irrelevant (name of the subject and so on)

```
###let us remove NA columns

datatrain <- rawdatatrain[,colSums(is.na(rawdatatrain))==0]





NeZ <- nearZeroVar(datatrain, saveMetrics=TRUE) ##let us remove Near zero variance variable
datatrain <- datatrain[,NeZ$nzv==FALSE]

datatrain <- datatrain[,-c(1:5)]  ###remove first 5 irrelevant column
```

Now that we have our dataset ready and more compact. We can do a partition of it in order to do an internal validation of our model.

```
## we are going to partition now, in order to have a training set and a cross validation set

set.seed(333)
inTrain <-  createDataPartition(y=datatrain$X..classe..., p=0.7, list=FALSE)
```

```
training <- datatrain[inTrain,]
valid <- datatrain[-inTrain,]

dim(training)
```

```
## [1] 13737    54
```

```
dim(valid)
```

```
## [1] 5885   54
```

We are finally ready now to try to use some predictive model.

Model1: Classification tree

Now we are going to try to fit a first model. In this part we are goint to use the calssification tree algorithm. We are going to introduce a k-fold control set to 10.

```
##first model
fitControl <- trainControl(method='cv', number = 10)
model1 <- train(
        X..classe...~ .,
        data=training,
        trControl=fitControl,
        method='rpart'
)


predictions <- predict(model1, newdata=valid)
confusionMatrix(predictions, valid$X..classe...)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ""A""" ""B""" ""C""" ""D""" ""E"""
##     ""A"""   1524    463    488    440    147
##     ""B"""     27    376     33    169    156
##     ""C"""    119    300    505    355    302
##     ""D"""      0      0      0      0      0
##     ""E"""      4      0      0      0    477
##
## Overall Statistics
##
##                Accuracy : 0.4897
##                  95% CI : (0.4769, 0.5026)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3331
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: ""A""" Class: ""B""" Class: ""C"""
## Sensitivity                 0.9104        0.33011        0.49220
```

```
## Specificity                 0.6348       0.91888       0.77856
## Pos Pred Value              0.4977       0.49409       0.31942
## Neg Pred Value              0.9469       0.85109       0.87895
## Prevalence                  0.2845       0.19354       0.17434
## Detection Rate              0.2590       0.06389       0.08581
## Detection Prevalence        0.5203       0.12931       0.26865
## Balanced Accuracy           0.7726       0.62450       0.63538
##                      Class: ""D""" Class: ""E"""
## Sensitivity                 0.0000       0.44085
## Specificity                 1.0000       0.99917
## Pos Pred Value                 NaN       0.99168
## Neg Pred Value              0.8362       0.88805
## Prevalence                  0.1638       0.18386
## Detection Rate              0.0000       0.08105
## Detection Prevalence        0.0000       0.08173
## Balanced Accuracy           0.5000       0.72001
```

We can see that the accuracy is really low (~0.5) and therefore the out of sample error (1-accuracy on predicted) is around 0.5 (which is relatively big). From there and by looking at the confusion matrix we can conclude that our first model does not seems to be accurate enough.

Model2: Random Forest

Now let us try to apply the Random Forest algorithm.

```
##first model


model2 <- randomForest(X..classe...~., data= training)


predictions2 <- predict(model2, newdata=valid)
confusionMatrix(predictions2, valid$X..classe...)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ""A""" ""B""" ""C""" ""D""" ""E"""
##     ""A"""   1674      1      0      0      0
##     ""B"""      0   1138      3      0      0
##     ""C"""      0      0   1023      8      0
##     ""D"""      0      0      0    956      5
##     ""E"""      0      0      0      0   1077
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9954, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: ""A""" Class: ""B""" Class: ""C"""
## Sensitivity                 1.0000        0.9991        0.9971
## Specificity                 0.9998        0.9994        0.9984
## Pos Pred Value              0.9994        0.9974        0.9922
## Neg Pred Value              1.0000        0.9998        0.9994
## Prevalence                  0.2845        0.1935        0.1743
## Detection Rate              0.2845        0.1934        0.1738
## Detection Prevalence        0.2846        0.1939        0.1752
## Balanced Accuracy           0.9999        0.9992        0.9977
##                      Class: ""D""" Class: ""E"""
## Sensitivity                 0.9917        0.9954
## Specificity                 0.9990        1.0000
## Pos Pred Value              0.9948        1.0000
## Neg Pred Value              0.9984        0.9990
## Prevalence                  0.1638        0.1839
## Detection Rate              0.1624        0.1830
## Detection Prevalence        0.1633        0.1830
## Balanced Accuracy           0.9953        0.9977
```

We can see that the accuracy is now way better (aournd 0.99, see above) and therefore the out of sample error is way smaller (therefore less than 0.01). Also we can check that the confusion matrix is given good prediction as well.

Model3: lda

Let's try to fit a last model to compare with our two precedent results. For the sake of the exercise more than for the accurate precise results, we are going to assumes a normal distribution for each variable, a variable mean which is spicific, and a common variance. Doing so, we can use the linear discriminant analysis (lda) method to fit our data. Let see what results do we get:

```r
model3 <- train(X..classe...~., training, method = "lda")
predictions3 <- predict(model3, newdata=valid)
confusionMatrix(predictions3, valid$X..classe...)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ""A""" ""B""" ""C""" ""D""" ""E"""
##     ""A"""   1388    150    103     41     34
##     ""B"""     53    740     99     30    174
##     ""C"""    103    148    671    119     76
##     ""D"""    123     57    120    736    115
##     ""E"""      7     44     33     38    683
##
## Overall Statistics
##
##                Accuracy : 0.7167
##                  95% CI : (0.705, 0.7282)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6418
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: ""A""" Class: ""B""" Class: ""C"""
## Sensitivity                 0.8292        0.6497        0.6540
## Specificity                 0.9221        0.9250        0.9082
## Pos Pred Value              0.8089        0.6752        0.6007
## Neg Pred Value              0.9314        0.9167        0.9255
## Prevalence                  0.2845        0.1935        0.1743
## Detection Rate              0.2359        0.1257        0.1140
## Detection Prevalence        0.2916        0.1862        0.1898
## Balanced Accuracy           0.8756        0.7873        0.7811
##                      Class: ""D""" Class: ""E"""
## Sensitivity                 0.7635        0.6312
## Specificity                 0.9157        0.9746
## Pos Pred Value              0.6394        0.8484
## Neg Pred Value              0.9518        0.9215
## Prevalence                  0.1638        0.1839
## Detection Rate              0.1251        0.1161
## Detection Prevalence        0.1956        0.1368
## Balanced Accuracy           0.8396        0.8029
```

We can see that in this case the accuracy is above the classification tree algorithm but we are below the accuracy of the random forest (and the other way around for the out of sample error). Therefore we are going to use the random forest algorithm that have a pretty good accuracy (and therefore a small out of sample error) to predict our final testing set.

Applying to the testing set.

We can now apply this model to the test set and we get the following prediction results

```
knitr::kable(data.frame(rawdatatest$X.,rawdatatest$X..user_name..,predict(model2,rawdatatest)))
```

rawdatatest.X.	rawdatatest.X..user_name..	predict.model2..rawdatatest.
“1	“”pedro“”	“”B“”“
“2	“”jeremy“”	“”A“”“
“3	“”jeremy“”	“”B“”“
“4	“”adelmo“”	“”A“”“
“5	“”eurico“”	“”A“”“
“6	“”jeremy“”	“”E“”“
“7	“”jeremy“”	“”D“”“
“8	“”jeremy“”	“”B“”“
“9	“”carlitos“”	“”A“”“
“10	“”charles“”	“”A“”“
“11	“”carlitos“”	“”B“”“
“12	“”jeremy“”	“”C“”“
“13	“”eurico“”	“”B“”“
“14	“”jeremy“”	“”A“”“
“15	“”jeremy“”	“”E“”“
“16	“”eurico“”	“”E“”“
“17	“”pedro“”	“”A“”“
“18	“”carlitos“”	“”B“”“
“19	“”pedro“”	“”B“”“
“20	“”eurico“”	“”B“”“

rawdatatest.X.	rawdatatest.X..user_name..	predict.model2..rawdatatest.