PML Course Project

VIboy

12 May 2016

Background

The goal of this project is to predict the manner in which 6 participants did the exercise described by the “classe” variable in the training set. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways according to the specification from A to E. We are free to use any of the other variables to predict with. In this report, I will describe how I built my model, how I used cross-validation, what I think the expected out of sample error is and why I made the choices that I did. Since the raw data consisted of 19622 observations of 160 variables, I decided to use only half the data set through sampling of 50% of the raw data set.

Loading the necessary software packages

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
library(caret)
library(dplyr)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.2.5
```

```
library(ggplot2)
```

Download the data

```
setwd("~/MachineLearning")
```

Load training data

The raw data was downloaded from the following url and saved to a data folder to be used in this project: fileUrl<-“https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)” download.file(fileUrl,destfile = “./data/projectPML.csv”)

```
rawTrain<-read.csv("data/projectPML.csv") #raw training set
dim(rawTrain)
```

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

Load testing data

Similarly, the testing data was downloaded from the following link and saved to a data subfolder to be used here. fileUrl1<-“https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)” download.file(fileUrl1,destfile = “./data/testPML.csv”)

```
rawTest<-read.csv("data/testPML.csv") #raw training set
dim(rawTest)
```

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

Sample raw data set

Since the raw data consist of 19622 observations, I decided to use just 9800 (50%) of these observations by sampling from the original data set and see how it performs in terms of accuracy. I used set.seed(123) so that the results can be repeated.

```
set.seed(123)
rawTrain1<- rawTrain[sample(1:nrow(rawTrain), 9800,replace=FALSE),]
dim(rawTrain1)
```

```
## [1] 9800  160
```

Remove data columns with more than 80% NAs.

We can see in the raw data that there are various columns that have a large number of NAs and so these have to be removed in both the training and test sets.

```
training <- rawTrain1[,colSums(is.na(rawTrain1)) < nrow(rawTrain1)*0.8]
dim(training) # 19622 93
```

```
## [1] 9800   93
```

```
testing <- rawTest[,colSums(is.na(rawTest)) < nrow(rawTest)*0.8]
dim(testing) # 20 60
```

```
## [1] 20 60
```

Remove “timestamp”, “window”, and “user” variables as well as factor variables

In both the training and testing data sets there are various 2-3 level factor variables that do not seem to have any impact on the manner in which the participants did the exercise according to the classe variable. These were removed so that what we finally have are variables with numeric and integer values for building the model.

```
training <- select(training, - contains("timestamp"), -ends_with("window"), -starts_with("user"), -X)
testing <- select(testing, - contains("timestamp"), -ends_with("window"), -starts_with("user"), -X)
```

Remove 2-3 level factor variables

```
training <- select(training, -contains("kurtosis"))
training <- select(training, -contains("skewness"))
training <- select(training, -contains("max"))
training <- select(training, -contains("min"))
training <- select(training, -contains("amplitude"))
dim(training)
```

```
## [1] 9800   53
```

```
dim(testing)
```

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

The above results now show that both the training and testing data sets have the same number of 53 predictor variables as recorded by the activity monitors in this exercise. This appear to be a good point from which to start building the model.

Cross-validation

For the purposes of cross-validation, the raw training set was split into a training and validation data sets according to the proportion of 70/30 for purposes of predicting performance and error rates of the various models before testing it on the test set. We intend to use 5-fold cross-validation in the random Forest algorith to build the model.

```
inTrain <- createDataPartition(y = training$classe, p=0.7, list=FALSE)
training <- training[inTrain,]
validation <- training[-inTrain,]
dim(training)
```

```
## [1] 6862   53
```

```
dim(validation)
```

```
## [1] 2066   53
```

Preprocessing with PCA

In order to check how many of the predictors are highly correlated, we used the following code:

```
M <- abs(cor(training[,-53]))
diag(M) <- 0
length(which(M>0.8))
```

```
## [1] 30
```

And it showed that 15 of these predictors were highly correlated since each correlated pair is counted twice. However, for this model where I am only using 50% of the observations, I would try to build the model without any preprocessing with Principal Components Analysis (preProcessing ="pca") in the training model. I would stick to this model without any preprocessing if the accuracy is more than 95%.

Using Random Forest to build the model

Since random Forest is one of the top performing algorithms in any prediction contest, I decided to use it to build our model to predict the 'classe' variable with preprocessing in the train function. For cross-validation a k-fold of 5 was used. Since the default ntree is 500, I decided to save time and reduce it to 100 hopefully without affecting the accuracy of the final model that is being built.

```
modelFit <- train(training$classe ~.,  method="rf", trControl=trainControl(method="cv", number = 5),ntree=100,
                  importance=TRUE, data=training)

modelFit
```

```
## Random Forest
##
## 6862 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5490, 5490, 5490, 5489, 5489
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9779935  0.9721588  0.010197210  0.012902251
##   27    0.9823661  0.9776923  0.005953536  0.007530663
##   52    0.9717281  0.9642401  0.005215572  0.006585650
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

The above results show an accuracy of 98.2% for the best model so the out-of-sample error is not expected to be less than 1.80%. We then proceeded to see how the final model will perform on the validation data set.

```
predRF <- predict(modelFit, validation)
confusionMatrix(predRF, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 600   0   0   0   0
##          B   0 380   0   0   0
##          C   0   0 346   0   0
##          D   0   0   0 336   0
##          E   0   0   0   0 404
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9982, 1)
##     No Information Rate : 0.2904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2904   0.1839   0.1675   0.1626   0.1955
## Detection Rate         0.2904   0.1839   0.1675   0.1626   0.1955
## Detection Prevalence   0.2904   0.1839   0.1675   0.1626   0.1955
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Surprisingly, the above results on the validation data set showed an accuracy of 100% which is better than that for the training set.

Answers for Test Set Predictions

The following code was used to do the predictions for the test set answers using the model built here:

```
answers <- predict(modelFit, testing)
answers
```

```
##  [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
```

These answers for the test set also gave a 100% accuracy in line with that of the validation set.

Conclusion:

This study on Human Activity Recognition (HAR) using a random Forest algorithm gave an accuracy ranging from 98.2 - 100% for the in-sample and out-of-sample data. We could achieve this by just using 50% of the raw data and 53 of the original 160 predictor variables recorded by the various activity monitors in this exercise. The model that was built was also able to achieve 100% correctness when it was tested against the testing set.