Practical Machine Learning Writeup

Amol Nankar

January 23, 2017

Background

In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information visit: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (look forsection on the Weight Lifting Exercise Dataset).

The training and test data for this project is downloaded from:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Project Goal

The goal of this Practical Machine Learning project writeup is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We may use any of the other variables to predict with. We should create a report describing how we built model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use our prediction model to predict 20 different test cases.

Data Cleansing and Manipulation

Read the source csv files and load them into data variable

```
library(caret)
```

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

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

```
pmlTrain <- read.csv("pml-training.csv", header=T, na.strings=c("NA", "#DIV/
0!"))
pmlTest <- read.csv("pml-testing.csv", header=T, na.strings=c("NA", "#DIV/0!"))
```

There being some NAs in the data, we select columns which don't contain NAs. Also, if we looke at data, it contains descriptive data which are not sensor values we need to filter out that data too using below code. Variables related to time and user information are excluded for a total of 51 variables and 19622 class measurements. Same variables are mainteined in test data set for predicting 20 test cases provided.

```
# exclude NA's from dataset
noNApmlTrain<-pmlTrain[, apply(pmlTrain, 2, function(x) !any(is.na(x)))]
dim(noNApmlTrain)
```

```
## [1] 19622    60
```

```
# variables with user information, time
cleanpmlTrain<-noNApmlTrain[,-c(1:8)]
dim(cleanpmlTrain)
```

```
## [1] 19622    52
```

```
# 20 test cases provided clean info - Validation data set
cleanpmltest<-pmlTest[,names(cleanpmlTrain[,-52])]
dim(cleanpmltest)
```

```
## [1] 20 51
```

Data Slicing or Partitioning

```
 # we are partitioning data to obtain a 75% training set and a 25% test set
inTrain<-createDataPartition(y=cleanpmlTrain$classe, p=0.75,list=F)
training<-cleanpmlTrain[inTrain,]
test<-cleanpmlTrain[-inTrain,]

#get dimensions for partitioned data
dim(training)
```

```
## [1] 14718    52
```

```
dim(test)
```

```
## [1] 4904   52
```

Prediction Process

Random forest trees were generated for training dataset. Then the generated algorithm was examnined under the partitioned training set to examine accuracy and estimated error of prediction. using 51 predictors for five classes using cross-validation at a 5-fold an accuracy of 99.2% with a 95% CI was achieved accompanied by a Kappa value of 0.99.

```
set.seed(100)
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rffit<-train(classe~.,data=training, method="rf", trControl=fitControl2, verbos
e=F)
```

```
## Loading required package: randomForest
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
predrf<-predict(rffit, newdata=test)
confusionMatrix(predrf, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    8    0    0    0
##          B    2  935    4    0    0
##          C    0    4  850   10    0
##          D    0    0    1  793    1
##          E    0    2    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9852   0.9942   0.9863   0.9989
## Specificity            0.9977   0.9985   0.9965   0.9995   0.9993
## Pos Pred Value         0.9943   0.9936   0.9838   0.9975   0.9967
## Neg Pred Value         0.9994   0.9965   0.9988   0.9973   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1907   0.1733   0.1617   0.1835
## Detection Prevalence   0.2857   0.1919   0.1762   0.1621   0.1841
## Balanced Accuracy      0.9981   0.9919   0.9953   0.9929   0.9991
```

```
pred20<-predict(rffit, newdata=cleanpmltest)
# Output for the prediction of the 20 cases provided
pred20
```

```
##  [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
```

A boosting algorithm was also run to confirm and be able to compare predictions.boosting approach presented less accuracy (96%). However, when the predictions for the 20 test cases were compared match was same for both ran algorimths.

```
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
gmbfit<-train(classe~.,data=training, method="gbm", trControl=fitControl2, verb
ose=F)
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnod
e = 10 on full training set
```

```
gmbfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 42 had non-zero influence.
```

```
class(gmbfit)
```

```
## [1] "train"         "train.formula"
```

```
predgmb<-predict(gmbfit, newdata=test)
confusionMatrix(predgmb, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1369   27    0    2    1
##          B   17  891   30    1    6
##          C    6   23  812   36    5
##          D    3    1   11  760   13
##          E    0    7    2    5  876
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.9542, 0.9653)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9494
##  Mcnemar's Test P-Value : 0.001439
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9814   0.9389   0.9497   0.9453   0.9723
## Specificity            0.9915   0.9863   0.9827   0.9932   0.9965
## Pos Pred Value         0.9786   0.9429   0.9206   0.9645   0.9843
## Neg Pred Value         0.9926   0.9853   0.9893   0.9893   0.9938
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2792   0.1817   0.1656   0.1550   0.1786
## Detection Prevalence   0.2853   0.1927   0.1799   0.1607   0.1815
## Balanced Accuracy      0.9864   0.9626   0.9662   0.9692   0.9844
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4139   69    0    0    2
##          B   29 2723   63    7   13
##          C   12   54 2475   78   18
##          D    4    0   25 2311   28
##          E    1    2    4   16 2645
##
## Overall Statistics
##
##                Accuracy : 0.9711
##                  95% CI : (0.9683, 0.9738)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9635
##  Mcnemar's Test P-Value : 1.413e-14
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9561   0.9642   0.9581   0.9775
## Specificity            0.9933   0.9906   0.9867   0.9954   0.9981
## Pos Pred Value         0.9831   0.9605   0.9386   0.9759   0.9914
## Neg Pred Value         0.9956   0.9895   0.9924   0.9918   0.9949
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2812   0.1850   0.1682   0.1570   0.1797
## Detection Prevalence   0.2860   0.1926   0.1792   0.1609   0.1813
## Balanced Accuracy      0.9911   0.9733   0.9754   0.9767   0.9878
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4139   69    0    0    2
##          B   29 2723   63    7   13
##          C   12   54 2475   78   18
##          D    4    0   25 2311   28
##          E    1    2    4   16 2645
##
## Overall Statistics
##
##                Accuracy : 0.9711
##                  95% CI : (0.9683, 0.9738)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9635
##  Mcnemar's Test P-Value : 1.413e-14
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9561   0.9642   0.9581   0.9775
## Specificity            0.9933   0.9906   0.9867   0.9954   0.9981
## Pos Pred Value         0.9831   0.9605   0.9386   0.9759   0.9914
## Neg Pred Value         0.9956   0.9895   0.9924   0.9918   0.9949
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2812   0.1850   0.1682   0.1570   0.1797
## Detection Prevalence   0.2860   0.1926   0.1792   0.1609   0.1813
## Balanced Accuracy      0.9911   0.9733   0.9754   0.9767   0.9878
```

Conclusion

We can conclude that we accurately predicted the classification of 20 observations using a Random Forest algorithm trained on a subset of data using less than 20% of the covariates.

The accuracy obtained (accuracy = 99.77%, and out-of-sample error = 0.23%) is obviously highly suspicious as it is never the case that machine learning algorithms are that accurate, and a mere 85% if often a good accuracy result.

Either 6 participants for whom we have data were extraordinarily obedient (for more than 19 thousand observations, a strong performance! This however might be explained by the highly controlled conditions of the data collection), or additional testing needs to be performed on other different participants.