Practical Machine Learning - Prediction Assignment Writeup

Ashish Veera

11/5/2017

Executive Summary

The objective and goal of this project is to predict the manner in which certain participants performed their exercise and machine learning classification of accelerometers data on the belt, forearm, arm, and dumbell of 6 participants.In training data “classe” is the outcome variable and uses certaing predictor variables inorder to predict 20 different test cases.The data for this project are retrieved from the below source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

The “classe” variable classifies the correct and incorrect outcomes of A, B, C, D, and E categories. Coursera project writeup describes the model cross validation and expected out of sample error rate.Random Forest Model has been applied successfully to predict all 20 different test cases.

Loading the data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

```
setwd("~/Desktop/PracticalMachineLearning")

if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists("./data/pml-training.csv")) {
  url.training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(url.training, destfile = "./data/pml-training.csv")
}

if (!file.exists("./data/pml-testing.csv")) {
  url.testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(url.testing, destfile = "./data/pml-testing.csv")
}
```

Reading and Processing data

```
train<- read.csv("./data/pml-training.csv")
test<- read.csv("./data/pml-testing.csv")

dim(train)
```

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

```
dim(test)
```

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

Note that both dataset are having the same variables (160 variables). Next step is to try remove the near zero variance variables or columns that contain N/A missing values.

```
train <- train[, colSums(is.na(train)) == 0]
test <- test[, colSums(is.na(test)) == 0]
classe <- train$classe
trainR <- grepl("^X|timestamp|window", names(train))
train <- train[, !trainR]
trainM <- train[, sapply(train, is.numeric)]
trainM$classe <- classe
testR <- grepl("^X|timestamp|window", names(test))
test<- test[, !testR]
testM <- test[, sapply(test, is.numeric)]
```

There were 107 variables with more than 95% of the data values missing. Those variables were removed from the data.

Data Partitioning

Partitioning Training data set into two data sets,70% for train data, 30% for test data as this will be used for cross validation purpose:

```
library(caret)
```

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

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

```
set.seed(12345)
inTrain <- createDataPartition(trainM$classe, p=0.70, list=F)
train_data <- trainM[inTrain, ]
test_data <- trainM[-inTrain, ]
```

Data Prediction and Modelling

Algorithm which will be used for the predictive model here is Random Forest

```
setting <- trainControl(method="cv", 5)
RandomForest <- train(classe ~ ., data=train_data, method="rf", trControl=setting,
ntree=250)
```

```
## 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
```

```
RandomForest
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9900996  0.9874758
##   27    0.9898812  0.9871998
##   52    0.9840573  0.9798299
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

We estimate the performance of the model built. Getting the accuracy as well as the estimated out-of-sample error.

```
    predict_RandomForest <- predict(RandomForest, test_data)
    confusionMatrix(test_data$classe, predict_RandomForest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   10 1125    4    0    0
##          C    0   14 1011    1    0
##          D    0    0   24  940    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9903
##                  95% CI : (0.9875, 0.9927)
##     No Information Rate : 0.286
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9877
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9868   0.9731   0.9958   1.0000
## Specificity            0.9998   0.9970   0.9969   0.9951   0.9994
## Pos Pred Value         0.9994   0.9877   0.9854   0.9751   0.9972
## Neg Pred Value         0.9976   0.9968   0.9942   0.9992   1.0000
## Prevalence             0.2860   0.1937   0.1766   0.1604   0.1833
## Detection Rate         0.2843   0.1912   0.1718   0.1597   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9969   0.9919   0.9850   0.9955   0.9997
```

```
accuracy <- postResample(predict_RandomForest, test_data$classe)
error<-1 - as.numeric(confusionMatrix(test_data$classe,
predict_RandomForest)$overall[1])
accuracy*100
```

```
## Accuracy    Kappa
## 99.03144 98.77458
```

```
error*100
```

```
## [1] 0.9685641
```

The accuracy of the model is 99.03% and the out-of-sample error is 0.9%

Predicting Results on the Test Data

Finally, will use the Random Forest model to predicy the results on the actual test data.

```
predictionRF<- predict(RandomForest, testM)
predictionRF
```

```
##  [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
```