Data Science: Practical Machine Learning - Project

Jagannatha Reddy

September 27, 2016

Problem Description

In this project we will be looking at a sensory data set from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the groupware website (see the section on the Weight Lifting Exercise Dataset). The training data for this project is downloaded from cloudfront website

In this project the following are the main focus items:

1. Cleaning the data so that only relevant predictors play a role in model creation
2. Build different models
3. Validate the models against the training set and observe how accurate they are
4. Predict the results for test set and validate the accuracy

Data Preparation

```
cache=TRUE
setwd("/Users/jagan/work/DataScience/PracticalMachineLearning/DataScience-PracticalMachineLearning-Proj
trainingFile <- "data/pml-training.csv"
testingFile  <- "data/pml-testing.csv"
if (!file.exists(trainingFile)) {
    dir.create("data", showWarnings = FALSE) #ignore Warning to recreate the directory
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile=trai
}

if (!file.exists(testingFile)) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile=testi
}
trainData <- read.csv(file=trainingFile, stringsAsFactors = FALSE)
testData  <- read.csv(file=testingFile, stringsAsFactors = FALSE)
dim(trainData)
```

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

As you can see we only have have 406 rows having the complete data but training set has 160 columns which is very high. Given this our goal is to identify only the columns which are relevant but at the same time have statistically significant data in generating the models. This means we have to selectively eliminate the columns without having to reduce the number of rows significantly

As a first step we will eliminate the near zero variance predictors using carat::nearZeroVar function as they will not play any significant role in coming up with the right model

```
library(lattice)
library(ggplot2)
library(caret)
```

```
NZVColumns <- nearZeroVar(trainData)
trainData  <- trainData[, -NZVColumns]
testData  <- testData[, -NZVColumns]
dim(trainData)
```

```
## [1] 19622   100
```

As you can see we could eliminate 60 columns from the dataset as these columns wouldn't impact the model generation. However we will have 100 columns which is very high in predicting the right model and hence there is further scope for cleaning up the data. As a next step we would eliminate the colums which have very sparse data

```
GoodColumn <- sapply(trainData, function(x) mean(!is.na(x))) > 0.95
trainData <- trainData[, GoodColumn==TRUE]
testData <- testData[, GoodColumn==TRUE]
dim(trainData)
```

```
## [1] 19622    59
```

```
trainData$classe <- as.factor(trainData$classe)
```

As you can see the first 5 columns (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp) shouldn't play a role in model building and hence eliminating these columns too

```
trainData <- trainData[, -c(1:5)]
testData <- testData[, -c(1:5)]
dim(trainData)
```

```
## [1] 19622    54
```

After this cleanup what we have left is with the complete cases and hence we can proceed with building the models

Build Models

In this section we would build models based on random forest ("rf"), generalized boosted regression ("gbm") and linear discriminant analysis ("lda") models.

```
set.seed(12345)
inTrain  = createDataPartition(trainData$classe, p = 0.70)[[1]]
training = trainData[inTrain,]
testing  = trainData[-inTrain,]

library(randomForest)
library(gbm)
library(lda)

rfModel  <- train(classe~., data=training, method="rf", verbose=FALSE)
gbmModel <- train(classe~., data=training, method="gbm", verbose=FALSE)
ldaModel <- train(classe~., data=training, method="lda", verbose=FALSE)
```

Model validation against training dataset

In this section we would validate our models against the training dataset.

```
rfPredictTrain  <- predict(rfModel, training)
confusionMatrix(rfPredictTrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
gbmPredictTrain <- predict(gbmModel, training)
confusionMatrix(gbmPredictTrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3901   11    0    0    0
##          B    4 2626   15    5    3
##          C    0   21 2377   22    1
##          D    1    0    3 2225   11
##          E    0    0    1    0 2510
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9929
##                  95% CI : (0.9913, 0.9942)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9880   0.9921   0.9880   0.9941
## Specificity            0.9989   0.9976   0.9961   0.9987   0.9999
## Pos Pred Value         0.9972   0.9898   0.9818   0.9933   0.9996
## Neg Pred Value         0.9995   0.9971   0.9983   0.9977   0.9987
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1912   0.1730   0.1620   0.1827
## Detection Prevalence   0.2848   0.1931   0.1762   0.1631   0.1828
## Balanced Accuracy      0.9988   0.9928   0.9941   0.9934   0.9970
```

```
ldaPredictTrain <- predict(ldaModel, training)
confusionMatrix(ldaPredictTrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3238  365  246  131  103
##          B  113 1757  220   83  360
##          C  239  316 1605  268  238
##          D  304  115  265 1684  224
##          E   12  105   60   86 1600
##
## Overall Statistics
##
##                Accuracy : 0.7195
##                  95% CI : (0.7119, 0.727)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.645
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8290   0.6610   0.6699   0.7478   0.6337
## Specificity            0.9140   0.9300   0.9064   0.9209   0.9765
## Pos Pred Value         0.7930   0.6936   0.6020   0.6497   0.8588
## Neg Pred Value         0.9308   0.9196   0.9286   0.9490   0.9221
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2357   0.1279   0.1168   0.1226   0.1165
## Detection Prevalence   0.2972   0.1844   0.1941   0.1887   0.1356
```

```
## Balanced Accuracy      0.8715   0.7955   0.7882   0.8344   0.8051
```

Let's list out the accuracies of these models against the training dataset

1. Accurary of randomforest model agaist training set: 1
2. Accurary of gbm model agaist training set: 0.992866
3. Accurary of lda model agaist training set: 0.7195166

```
rfModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.17%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    5 2652    1    0    0 0.0022573363
## C    0    5 2391    0    0 0.0020868114
## D    0    0    8 2244    0 0.0035523979
## E    0    0    0    3 2522 0.0011881188
```

Based on this information its very clear that randomforest & gbm models accuracy is much superior than lda model. Looking at the Out-of-Bag (OOB) error of randomforest model its very clear that its an accurate model. However the complexity in building the randomforest model is higher compared to the other 2 models. When the prediction is built using tree model (method=rpart) the accuracy wasn't that good and not reported here.

Apply the model to the test dataset

In this section we would validate our models against the training dataset.

```
rfPredict  <- predict(rfModel, testing)
confusionMatrix(rfPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    4    0    0
##          C    0    1 1022    7    0
##          D    0    0    0  957    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9964
##                  95% CI : (0.9946, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9961   0.9927   0.9963
## Specificity            0.9988   0.9992   0.9984   0.9992   1.0000
## Pos Pred Value         0.9970   0.9965   0.9922   0.9958   1.0000
## Neg Pred Value         1.0000   0.9987   0.9992   0.9986   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1737   0.1626   0.1832
## Detection Prevalence   0.2853   0.1932   0.1750   0.1633   0.1832
## Balanced Accuracy      0.9994   0.9969   0.9972   0.9960   0.9982
```

```
gbmPredict <- predict(gbmModel, testing)
confusionMatrix(gbmPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    9    0    0    0
##          B    3 1117   20    7    3
##          C    0   12 1004   13    0
##          D    0    1    2  944   11
##          E    0    0    0    0 1068
##
## Overall Statistics
##
##                Accuracy : 0.9862
##                  95% CI : (0.9829, 0.9891)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9826
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9807   0.9786   0.9793   0.9871
## Specificity            0.9979   0.9930   0.9949   0.9972   1.0000
## Pos Pred Value         0.9946   0.9713   0.9757   0.9854   1.0000
## Neg Pred Value         0.9993   0.9954   0.9955   0.9959   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1898   0.1706   0.1604   0.1815
## Detection Prevalence   0.2855   0.1954   0.1749   0.1628   0.1815
## Balanced Accuracy      0.9980   0.9869   0.9867   0.9882   0.9935
```

```
ldaPredict <- predict(ldaModel, testing)
confusionMatrix(ldaPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1404  168   92   52   44
##          B   39  728   99   47  166
##          C  110  146  655  133   89
##          D  112   45  144  694  103
##          E    9   52   36   38  680
##
## Overall Statistics
##
##                Accuracy : 0.7071
##                  95% CI : (0.6952, 0.7187)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6291
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8387   0.6392   0.6384   0.7199   0.6285
## Specificity            0.9155   0.9260   0.9016   0.9179   0.9719
## Pos Pred Value         0.7977   0.6747   0.5781   0.6321   0.8344
## Neg Pred Value         0.9345   0.9145   0.9219   0.9436   0.9207
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2386   0.1237   0.1113   0.1179   0.1155
## Detection Prevalence   0.2991   0.1833   0.1925   0.1866   0.1385
## Balanced Accuracy      0.8771   0.7826   0.7700   0.8189   0.8002
```

Let's list out the accuracies of these models against the test dataset

1. Accurary of randomforest model agaist training set: 0.9964316
2. Accurary of gbm model agaist training set: 0.9862362
3. Accurary of lda model agaist training set: 0.7070518

Executive Summary

Based on the analysis done on the activity data it is evident that **Random forest** model outperforms the **gbm** and **lda** models in terms of accuracy. However gbm model's performance is also very close to that of randomforest but not the better. These conclusions are based on the statistically significant dataset and hence the confidence is also higher