Practical Machine Learning Project

(Part of Coursera - Johns Hopkins University - Data Science Specialization)

Author: Aled Evans

Introduction

This assignment focuses on fitness data gathered from accelerometers attached to different parts of the body. Using the data the assignment applies different machine learning models and identifies the best model for predicting the "classee" variable from a separate test dataset.

The data was downloaded from: http://groupware.les.inf.puc-rio.br/har. (Human Activity recognition, Groupware @ LES - Weight Lifting Exercises Dataset).

The Random Forest Model was selected (with the Generalized Boosted Model a very close second). The resulting predictions for "classe" in the test set scored 20 out of 20 - a 100% prediction success.

Loading Data and Preparation

```
# Prepare the R environment by loading required packages
library(plyr)
library(caret)
library(randomForest)
library(gbm)
library(survival)
library(kernlab)
library(splines)
library(parallel)
library(e1071)
# set seed for reproducibility
set.seed(425)
```

Loading Data for training

```
 # Loading Data from url
urlTraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(urlTraining, destfile="pml-training.csv")
dataTraining1 <- read.csv("pml-training.csv", header=TRUE)
```

Exploratory Data Analysis 1 - Review datasets

```
# use "dim" to give brief overveiw of loaded dataset structure
dim(dataTraining1)
```

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

The dataset has 160 variables and 19622 observations (rows).

From use of "str" we see a number of NA values in a number of variables (just from reviewing the first 40 rows) [The code and printout is in Appendix A]. To increase the speed and efficiency of the training, a number or variables are removed. Variables with a large number of NA values (greater than 90%) are removed and also the first column - a variable that is irrelevant to the training model.

```
# remove number column - write to "dataTraining" dataframe for further processing
dataTraining <- dataTraining1[,-1]
# Remove varables with NA greater than 90%. "dim" to review datset structure
naVar <- sapply(dataTraining, function(x) mean(is.na(x))) > 0.90
dataTraining <- dataTraining[, naVar==FALSE]
dim(dataTraining)
```

```
## [1] 19622    92
```

There are now 92 varables. A further processing procedure is to remove variables with Near Zero Variance.

```
# remove variables with Near Zero Variance
nearZero <- nearZeroVar(dataTraining)
dataTraining <- dataTraining[, -nearZero]
dim(dataTraining)
```

```
## [1] 19622    58
```

After data processing there are now 58 variables. The reduction in dataset size will significantly improve the training speed without impacting on the accuracy of the models.

Data Splitting For Training

For the training of the models, the dataset is partitioned (split) 75% for training and 25% for cross-validation. (The datasets are given 'tidy' in their name to indicate they have been completed processed.)

```
# split dataframe training and for cross-validation.
dataTraining2 <- createDataPartition(y=dataTraining$classe,p=0.75, list=FALSE)
tidyTraining <- dataTraining[dataTraining2,]
tidyCrossVal <- dataTraining[-dataTraining2,]
# check structure of the partitioned datesets.
dim(tidyTraining)
```

```
## [1] 14718    58
```

```
dim(tidyCrossVal)
```

```
## [1] 4904   58
```

Model Selection

Three models are trained and assessed - the best performing model will then be selected for the prediction stage. The models are - 1 Random Forest; 2- Support Vector Machines with Linear Kernel; and 3 - Generalized Boosted Model. 'kappa' is also included for use as a metric is assessing the best model. A confusion matrix is also generated for each model to aid in model assessment.

1 - Random Forest Model

```r
# set a control group - use 6 'k' fold
rfControl <- trainControl(method = "oob", number = 6)
# Random forest Model - use 300 trees
modelRandomF <- train(classe ~., data=tidyTraining, method="rf", ntree=300, metric="Kappa", trControl=r
modelRandomF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 40
##
##         OOB estimate of  error rate: 0.05%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4185    0    0    0    0 0.0000000000
## B    1 2847    0    0    0 0.0003511236
## C    0    2 2563    2    0 0.0015582392
## D    0    0    1 2410    1 0.0008291874
## E    0    0    0    0 2706 0.0000000000
```

```r
# cross validation of random forest model
predictRandomF <- predict(modelRandomF, tidyCrossVal)
confMatxRandomF <-confusionMatrix(predictRandomF, tidyCrossVal$classe)
confMatxRandomF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    1    0    0
##          C    0    1  854    0    0
##          D    0    0    0  804    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9979, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9988   1.0000   0.9989
## Specificity            0.9997   0.9997   0.9998   0.9998   1.0000
## Pos Pred Value         0.9993   0.9989   0.9988   0.9988   1.0000
```

```
## Neg Pred Value         1.0000   0.9995   0.9998   1.0000   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1741   0.1639   0.1835
## Detection Prevalence   0.2847   0.1933   0.1743   0.1642   0.1835
## Balanced Accuracy      0.9999   0.9988   0.9993   0.9999   0.9994
```

2 - Support Vector Machines Model - with linear kernal

```
# train SVMlinear
modelSVM = train(classe ~., data=tidyTraining, method="svmLinear", metric="Kappa")
modelSVM$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 3887
##
## Objective Function Value : -1062.338 -181.6431 -4.5 -0.478 -1044.128 -50.4787 -1.1213 -608.2038 -38.
## Training error : 0.082416
```

```
# SVM cross validation
predictSVM <- predict(modelSVM, tidyCrossVal)
confMatxSVM <- confusionMatrix(predictSVM, tidyCrossVal$classe)
confMatxSVM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1357  115    4    0    0
##          B   30  770   58    4    1
##          C    8   64  782   61    1
##          D    0    0   11  690   41
##          E    0    0    0   49  858
##
## Overall Statistics
##
##                Accuracy : 0.9088
##                  95% CI : (0.9004, 0.9168)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8845
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9728   0.8114   0.9146   0.8582   0.9523
```

```
## Specificity            0.9661   0.9765   0.9669   0.9873   0.9878
## Pos Pred Value         0.9194   0.8922   0.8537   0.9299   0.9460
## Neg Pred Value         0.9889   0.9557   0.9817   0.9726   0.9892
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2767   0.1570   0.1595   0.1407   0.1750
## Detection Prevalence   0.3010   0.1760   0.1868   0.1513   0.1850
## Balanced Accuracy      0.9694   0.8939   0.9408   0.9228   0.9700
```

3 - Generalized Boosted Model (GBM)

```
# set control
gbmControl <- trainControl(method = "repeatedcv")
# train GBM
modelGBM <- train(classe ~., data=tidyTraining, method="gbm", metric="Kappa", trControl=gbmControl, verb
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 79 predictors of which 44 had non-zero influence.
```

```
# cross validation of GBM
predictGBM <- predict(modelGBM, tidyCrossVal)
confMatxGBM <-confusionMatrix(predictGBM, tidyCrossVal$classe)
confMatxGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
##          B    0  942    1    0    0
##          C    0    1  847    0    0
##          D    0    0    7  804    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9906   1.0000   0.9978
## Specificity            0.9983   0.9997   0.9998   0.9978   1.0000
## Pos Pred Value         0.9957   0.9989   0.9988   0.9889   1.0000
## Neg Pred Value         1.0000   0.9982   0.9980   1.0000   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
```

```
## Detection Rate         0.2845   0.1921   0.1727   0.1639   0.1833
## Detection Prevalence   0.2857   0.1923   0.1729   0.1658   0.1833
## Balanced Accuracy      0.9991   0.9962   0.9952   0.9989   0.9989
```

Results

Random Forest - 99.92% accuracy. Kappa = 0.999

SVM - 90.88 accuracy. Kappa = 0.8845

GBM - 99.65% accuracy. Kappa = 0.9956

It is very close between Random Forest and GBM. (SVM is not as strong). Random Forest has better accuracy and kappa (though the difference is very small) so Random Forest will be selected to predict the testing dataset.

Apply Random Forest Model to test dataset for "classe" prediction.

Load test data

```
# Load test data from URL
urlTesting <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(urlTesting, destfile="pml-testing.csv")
testing <- read.csv("pml-testing.csv", header=TRUE)
# Check dimension of testing dataset
dim(testing)
```

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

Apply Random Forest Model to test case "testing" for the 20 different test cases. (As the dataset is much smaller, the pre-processing carried out for the training set is not applied.)

```
# apply Random Forest Model model to testing dataset
predictTest <- predict(modelRandomF, newdata=testing)
# print out prediction
print(as.data.frame(predictTest))
```

```
##    predictTest
## 1            B
## 2            A
## 3            B
## 4            A
## 5            A
## 6            E
## 7            D
## 8            B
## 9            A
## 10           A
## 11           B
## 12           C
## 13           B
## 14           A
```

```
## 15           E
## 16           E
## 17           A
## 18           B
## 19           B
## 20           B
```

Conclusion

The resulting predictions for "classe" in the test set is 20 out of 20 (when tested on the Coursera website). A 100% prediction success.

Both the Random Forest and GBM proved to be very strong models for the task. The out of sample error rate is very for both models (under 0.005%)

Appendix A

Output of "str" of original training dataset when loaded. Used in the exploratory data analysis stage.

```
str(dataTraining1, list.len=40)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##   [list output truncated]
```