Practical Machine Learning Project - Predict Weightlifting Fashion

Michael Lioznov

Synopsis

Goal of this project is to devise prediction model for predicting the way in which weightlifting exercise participants perform barbell lifts based on data from accelerometers on their belt, forearm, arm, and dumbbell. Participants can perform barbell lifts correctly and incorrectly in 5 different ways:

Class A - correctly per specification.

Class B - throwing the elbows to the front.

Class C - lifting the dumbbell only halfway.

Class B - lowering the dumbbell only halfway.

Class E - throwing the hips to the front.

More information is available here: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises

Data Load and Preparation

Load required packages

```
# load the required packages
suppressMessages(library(caret));
```

```
## Warning: package 'caret' was built under R version 3.4.1
```

```
## Warning: package 'ggplot2' was built under R version 3.4.1
```

```
suppressMessages(library(rattle));
```

```
## Warning: package 'rattle' was built under R version 3.4.1
```

```
suppressMessages(library(rpart));
suppressMessages(library(rpart.plot))
```

```
## Warning: package 'rpart.plot' was built under R version 3.4.1
```

```
suppressMessages(library(randomForest));
```

```
## Warning: package 'randomForest' was built under R version 3.4.1
```

Load training and testing data

```
# load training and testing data
setwd("C:/Users/egdbb48/PracticalMachineLearning/CourseProject")
download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",destfile="Trai
download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",destfile="Testi
testing <- read.csv("TestingData.csv", na.strings = c("NA", "" ,"#DIV/0!"), header = TRUE)
training <- read.csv("TrainingData.csv", na.strings = c("NA", "", "#DIV/0!"), header = TRUE)
```

Exploration of the training and testing sets shows a lot of undefined variables that cannot and should not be used as predictors; also the first 7 variables (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window) obviously cannot correllate with the weightlifting mannors demonstrated by participants.

Cleansing datasets from nonsensical fields:

```
# Clean test set first - to limit useful predictors:
testClean <- testing[,colSums(is.na(testing)) == 0 ][, -c(1:7)] # & !(colnames(testing) %in% c("problem
# Find which predictors in the training set should be chosen for modeling mannor of weightlifting (clas
ColNames <- names(testClean[, !(colnames(testClean) %in% c("problem_id"))])
# Subset the training set
trainClean <- training[ ,c(ColNames,"classe")] # tail(trainClean)
#names(trainClean)
```

Data partitioning

To tune and compare possible models before using them on the testing (testClean) data set the training set was split into the training (Train) and validation (Validate) data sets

```
set.seed(4321)
PartitionFunc <- createDataPartition(y=trainClean$classe, p=3/4, list=FALSE)
Train <- trainClean[PartitionFunc, ]
Validate <- trainClean[-PartitionFunc, ]
```

Compare Prediction Algorithms

Random Forests (rf) and Classification Trees (rpart) were compared for accuracy.

Classification Trees

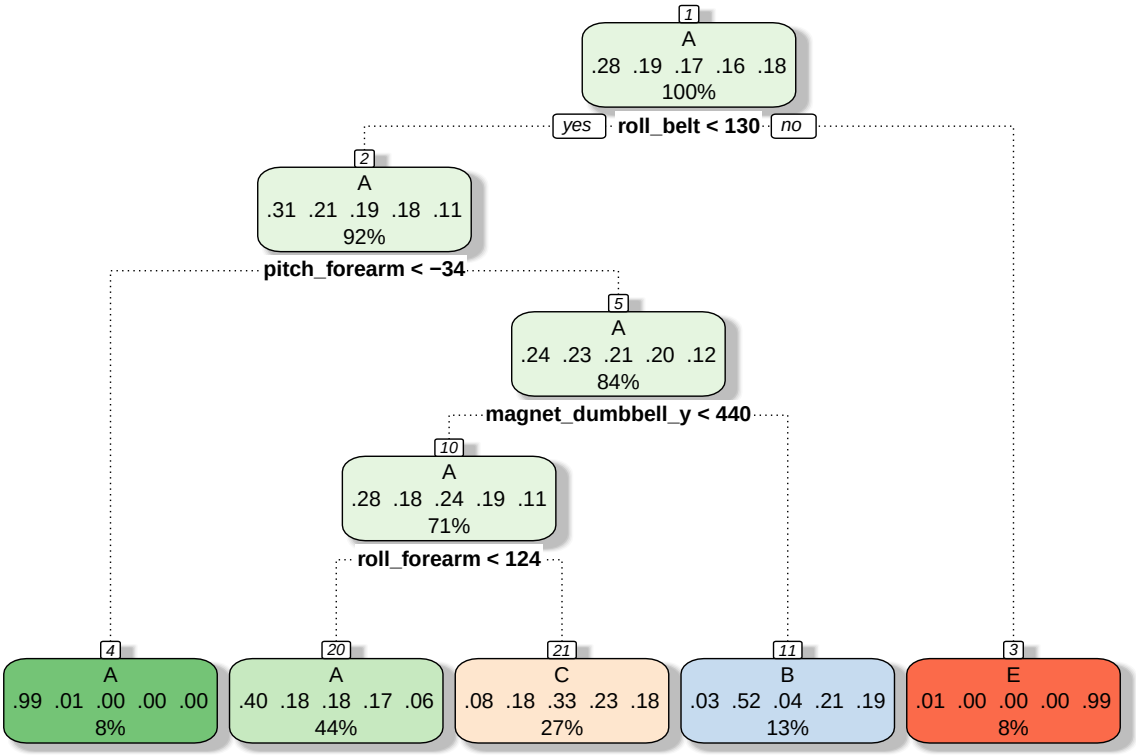
For Classification Trees the 20 - fold cross validation was used:

```
contr <- trainControl(method = "cv", number = 20)
Modelrpart <- train(classe ~ ., data = Train, method = "rpart",trControl = contr)
print(Modelrpart, digits = 4)
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (20 fold)
## Summary of sample sizes: 13981, 13981, 13983, 13982, 13982, 13983, ...
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03503  0.5032    0.35086
##   0.06038  0.4262    0.22666
##   0.11488  0.3225    0.05824
```

```
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03503.
```

```
fancyRpartPlot(Modelrpart$finalModel)
```



Rattle 2017−Sep−03 11:33:56 egdbb48

```
# Run prediction and show results on Validation set
PredictOnValidateRPART <- predict(Modelrpart, Validate)
confusionMatrix(Validate$classe, PredictOnValidateRPART)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1272   19  102    0    2
##          B  402  312  235    0    0
##          C  390   37  428    0    0
##          D  327  166  311    0    0
##          E  119  125  248    0  409
##
## Overall Statistics
##
##                Accuracy : 0.4937
##                  95% CI : (0.4796, 0.5078)
##     No Information Rate : 0.5118
##     P-Value [Acc > NIR] : 0.9947
##
```

```
##                   Kappa : 0.3389
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5068  0.47344  0.32326       NA  0.99513
## Specificity            0.9486  0.84994  0.88073   0.8361  0.89050
## Pos Pred Value         0.9118  0.32877  0.50058       NA  0.45394
## Neg Pred Value         0.6472  0.91226  0.77871       NA  0.99950
## Prevalence             0.5118  0.13438  0.26998   0.0000  0.08381
## Detection Rate         0.2594  0.06362  0.08728   0.0000  0.08340
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7277  0.66169  0.60199       NA  0.94282
```

The prediction attempt on the Validation set shows unsatisfactory accuracy of 0.4936786 and out-of-sample error rate of 0.5063214

Random Forests

```
Modelrf <- train(classe ~ .,data=Train,method='rf',ntree=100)
print(Modelrf, digits = 4)
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9887    0.9857
##   27    0.9881    0.9850
##   52    0.9791    0.9735
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

The following prediction attempt on the Validation set shows great accuracy:

```
# Run prediction and show results on Validation set
PredictOnValidateRf <- predict(Modelrf, Validate)
confusionMatrix(Validate$classe, PredictOnValidateRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    9  934    6    0    0
##          C    0   14  840    1    0
```

```
##          D    0    0   12  791    1
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9906
##                  95% CI : (0.9875, 0.9931)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9881
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9936   0.9852   0.9790   0.9950   0.9989
## Specificity            1.0000   0.9962   0.9963   0.9968   0.9993
## Pos Pred Value         1.0000   0.9842   0.9825   0.9838   0.9967
## Neg Pred Value         0.9974   0.9965   0.9956   0.9990   0.9998
## Prevalence             0.2863   0.1933   0.1750   0.1621   0.1833
## Detection Rate         0.2845   0.1905   0.1713   0.1613   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9968   0.9907   0.9877   0.9959   0.9991
```

Conclusion

Prediction accuracy of 0.9906199 is achieved by using RandomForests algorithm. The out-of-sample error rate for the Random Forests algorithm is 0.0093801. Although many predictors seem highly correlated, Random Forests algorithm chooses a subset of predictors at each split. This leads to high accuracy.

Random Forests algorythm was used for predicting weightlifting fashions in the testing set

```
# TestPrediction <- predict(Modelrf, newdata=testClean)
# TestPredictionResults <- data.frame(problem_id=testClean$problem_id,predicted=TestPrediction)
# print(TestPredictionResults)
```