Machine Learning - Course Project

SB

August 16, 2014

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:
Class A: Exactly according to the specification
Class B: Throwing the elbows to the front
Class C: Lifting the dumbbell only halfway
Class D: Lowering the dumbbell only halfway
Class E: Throwing the hips to the front

Training Data has been provided to reflect the resulting 'Class' of each observation from the above exercise performed by the six participants. The goal is to determine the best 'Prediction Model', which can be used to predict the 'Class' for the Test data provided separately

Synopsis:

- Since the goal of the exercise is to classify the Test data into one of the five 'Classes' defined in the Training data, different classification models have been used to test the accuracy of the models
- The Training data will be further divided into 'Training' and 'Testing' sets in the ratio of 70/30. Each classification model will be run against the 'Training' set and cross-validated against the 'Testing' set. The model with highest accuracy will be selected to predict the 'classe' for the Testing Data provided.

Data Processing:

- Download the files to the working directory

```
if(!file.exists("training.csv")) {
    fileTrain<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
    download.file(fileTrain, destfile= "./training.csv", method="curl")
}

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

- Load the data into Training and Testing data frames

```
trainingData<- read.table("training.csv", header=TRUE, sep=",")
testingData<- read.table("testing.csv", header=TRUE, sep=",")
```

- Feature Selection: Since the model will finally be run against the Testing Data, summarize the Testing

Data and create a sub-set Training Data set and Testing Data set with only numerical columns not missing majority of the values (NAs)

```
trainSet <- trainingData[,c("roll_belt","pitch_belt","yaw_belt", "total_accel_belt", "gyros
_belt_x","gyros_belt_y","gyros_belt_z","accel_belt_x","accel_belt_y","accel_belt_z","magnet
_belt_x","magnet_belt_y","magnet_belt_z","roll_arm","pitch_arm","yaw_arm","total_accel_arm",
"gyros_arm_x","gyros_arm_y","gyros_arm_z","accel_arm_x","accel_arm_y","accel_arm_z","magnet
_arm_x","magnet_arm_y","magnet_arm_z","roll_dumbbell","pitch_dumbbell","yaw_dumbbell","tota
l_accel_dumbbell","gyros_dumbbell_x","gyros_dumbbell_y","gyros_dumbbell_z","accel_dumbbell_
x","accel_dumbbell_y","accel_dumbbell_z","magnet_dumbbell_x","magnet_dumbbell_y","magnet_du
mbbell_z","roll_forearm","pitch_forearm","yaw_forearm","total_accel_forearm","gyros_forearm
_x","gyros_forearm_y","gyros_forearm_z","accel_forearm_x","accel_forearm_y","accel_forearm_
z","magnet_forearm_x","magnet_forearm_y","magnet_forearm_z", "classe")]
testSet <- testingData[,c("roll_belt","pitch_belt","yaw_belt", "total_accel_belt", "gyros_b
elt_x","gyros_belt_y","gyros_belt_z","accel_belt_x","accel_belt_y","accel_belt_z","magnet_b
elt_x","magnet_belt_y","magnet_belt_z","roll_arm","pitch_arm","yaw_arm","total_accel_arm","
gyros_arm_x","gyros_arm_y","gyros_arm_z","accel_arm_x","accel_arm_y","accel_arm_z","magnet_
arm_x","magnet_arm_y","magnet_arm_z","roll_dumbbell","pitch_dumbbell","yaw_dumbbell","total
_accel_dumbbell","gyros_dumbbell_x","gyros_dumbbell_y","gyros_dumbbell_z","accel_dumbbell_x"
,"accel_dumbbell_y","accel_dumbbell_z","magnet_dumbbell_x","magnet_dumbbell_y","magnet_dumb
bell_z","roll_forearm","pitch_forearm","yaw_forearm","total_accel_forearm","gyros_forearm_x"
,"gyros_forearm_y","gyros_forearm_z","accel_forearm_x","accel_forearm_y","accel_forearm_z",
"magnet_forearm_x","magnet_forearm_y","magnet_forearm_z")]
```

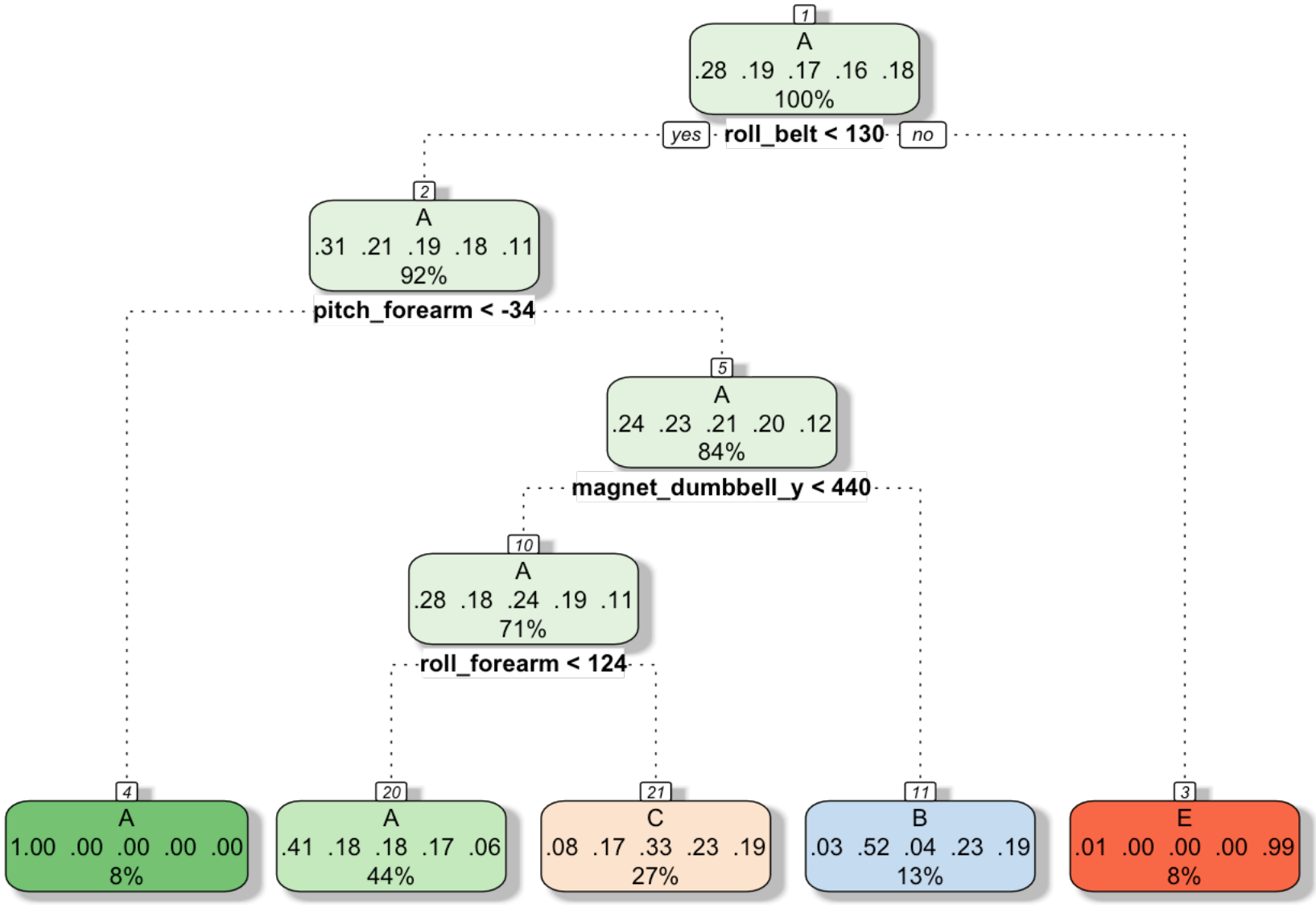
- Load the required libraries. Divide the trainingData into Training and Test sets in 70/30 ratio.

```
library(caret); library(rpart); library(randomForest); library(psych); library(rattle)
inTrain <- createDataPartition (y= trainSet$classe, p=0.7, list = FALSE)
trainSub <- trainSet[inTrain,]
testSub <- trainSet[-inTrain,]
```

Recursive Partitioning Method:

- Train the dataset using the ‘Recursive Partitioning’ method. Show the resulting classification tree

```
modelRpart <- train(classe~., method="rpart", data=trainSub)
fancyRpartPlot(modelRpart$finalModel)
```



Rattle 2014-Aug-17 18:22:48 ls1518

Expected 'Out of Sample Error' of this model will be (1-Accuracy) of the row with least Complexity Parameter (cp) value shown below

```
modelRpart
```

```
## CART
##
## 13737 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   cp    Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.04  0.5       0.4    0.01         0.02
##   0.06  0.4       0.2    0.07         0.1
##   0.1   0.3       0.09   0.04         0.05
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.04.
```

- Predict the values for 'Classe' variable by applying the trained model to the test sub-set.

```
predRpart <- predict(modelRpart,testSub)
```

'Estimated Error' of this model based on the cross-validation with the Test subset is represented by (1-Overall Accuracy) shown in the resuls of the Confusion Matrix below:

```
confusionMatrix(predRpart, testSub$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1542  488  482  446  153
##          B   28  375   46  168  161
##          C   99  276  498  350  259
##          D    0    0    0    0    0
##          E    5    0    0    0  509
##
## Overall Statistics
##
##                Accuracy : 0.497
##                  95% CI : (0.484, 0.51)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.342
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.921   0.3292   0.4854    0.000   0.4704
## Specificity             0.627   0.9151   0.7975    1.000   0.9990
## Pos Pred Value          0.496   0.4820   0.3360      NaN   0.9903
## Neg Pred Value          0.952   0.8504   0.8801    0.836   0.8933
## Prevalence              0.284   0.1935   0.1743    0.164   0.1839
## Detection Rate          0.262   0.0637   0.0846    0.000   0.0865
## Detection Prevalence    0.529   0.1322   0.2518    0.000   0.0873
## Balanced Accuracy       0.774   0.6222   0.6414    0.500   0.7347
```

Random Forest Method:

- Train the dataset using the ‘Random Forest’ method. To avoid long computational times, we are limiting the number of cross-validations to 5 through TrainControl parameter

```
fitControl = trainControl(method = "cv", number = 5)
modelRf <- train(classe~., method="rf", trControl=fitControl, data=trainSub, prox=TRUE)
```

Expected ‘Out of Sample Error’ of this model will be (1-Accuracy) of the row with least Complexity Parameter (cp) value shown below

```
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10990, 10990, 10989, 10990, 10989
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     1         1      0.001        0.002
##   30    1         1      0.002        0.003
##   50    1         1      0.003        0.004
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

- Predict the values for 'Classe' variable by applying the trained model to the test sub-set.

```
predRf <- predict(modelRf,testSub)
```

'Estimated Error' of this model based on the cross-validation with the Test subset is represented by (1-Overall Accuracy) shown in the resuls of the Confusion Matrix below:

```
confusionMatrix(predRf, testSub$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   11    0    0    0
##          B    1 1126    4    1    1
##          C    0    2 1019    4    1
##          D    0    0    3  958    0
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.995
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.989    0.993    0.994    0.998
## Specificity             0.997    0.999    0.999    0.999    1.000
## Pos Pred Value          0.993    0.994    0.993    0.997    0.999
## Neg Pred Value          1.000    0.997    0.999    0.999    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.191    0.173    0.163    0.184
## Detection Prevalence    0.286    0.193    0.174    0.163    0.184
## Balanced Accuracy       0.998    0.994    0.996    0.997    0.999
```

Boosting Method:

- Train the dataset using the ‘Boosting’ method. To avoid long computational times, we are limiting the number of cross-validations to 5 through Train Control parameter

```
fitControl = trainControl(method = "cv", number = 5)
modelGbm <- train(classe ~., method="gbm", trControl=fitControl, data=trainSub, verbose=FAL
SE)
```

Expected ‘Out of Sample Error’ of this model will be (1-Accuracy) of the row with least Complexity Parameter (cp) value shown below

```
modelGbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10989, 10989, 10990, 10991, 10989
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa  Accuracy SD  Kappa SD
##   1                  50       0.8       0.7    0.02         0.02
##   1                  100      0.8       0.8    0.008        0.01
##   1                  200      0.9       0.8    0.009        0.01
##   2                  50       0.9       0.8    0.01         0.01
##   2                  100      0.9       0.9    0.007        0.008
##   2                  200      0.9       0.9    0.004        0.005
##   3                  50       0.9       0.9    0.008        0.01
##   3                  100      0.9       0.9    0.004        0.005
##   3                  200      1         0.9    0.002        0.002
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3 and shrinkage = 0.1.
```

- Predict the values for ‘Classe’ variable by applying the trained model to the test sub-set.

```
predGbm <- predict(modelGbm,testSub)
```

‘Estimated Error’ of this model based on the cross-validation with the Test subset is represented by (1-Overall Accuracy) shown in the resuls of the Confusion Matrix below:

```
confusionMatrix(predGbm, testSub$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   25    0    1    1
##          B   16 1080   36    3   11
##          C    4   30  976   26    5
##          D    4    2   10  932   11
##          E    1    2    4    2 1054
##
## Overall Statistics
##
##                Accuracy : 0.967
##                  95% CI : (0.962, 0.971)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.958
##  Mcnemar's Test P-Value : 0.00167
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.985    0.948    0.951    0.967    0.974
## Specificity             0.994    0.986    0.987    0.995    0.998
## Pos Pred Value          0.984    0.942    0.938    0.972    0.992
## Neg Pred Value          0.994    0.988    0.990    0.994    0.994
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.280    0.184    0.166    0.158    0.179
## Detection Prevalence    0.285    0.195    0.177    0.163    0.181
## Balanced Accuracy       0.989    0.967    0.969    0.981    0.986
```

Conclusions:

- As shown in the results of the confusionMatrix for each of the models, Random Forest method offers the highest accuracy, followed by Boosting method. Lowest accuracy is offered by the simple ‘Recursive Partitioning’ method out of the three models.
- Based on these results, we choose to use the Random Forest model to predict the results for the Testing Data

```
predVal <- predict(modelRf, testSet)
predVal
```

```
##  [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
```