Practical Machine Learning Project

Tiblez

11/2/2016

Introduction

This project uses data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to investigate how well people do a particular activity. These participants perform barbell lifts correctly and incorrectly in the following 5 different ways.
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 halfwa and
Class E - throwing the hips to the front
Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.
Our goal is to predict the manner in which these 6 participants did the exercise. This report describes how the prediction model is built, how cross validation is used, what the expected out of sample error is, and why the choices for developing the model are done. The developed model was also used to predict 20 different test cases.

Data Processing and Cleaning

The data for this project is aquired from the Weight Lifting Exercise Dataset (http://groupware.les.inf.puc-rio.br/har).
The training data 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)

```
training<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trai
ning.csv"),
                    na.string=c("NA","","#DIV/0!"))
testing<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testi
ng.csv"),
                   na.string=c("NA","","#DIV/0!"))
```

The training dataset contains 19622 observations of 160 variables and the testing dataset contains 20 observations of 160 variables.

Data Cleaning

The first seven varibales, which are identifiers, are removed from the datasets since they are not needed for predection. Furthermore, columns that contain missing values are also removed from both the training and testing datasets.

```
training<- training[,-c(1:7)]
testing<- testing[,-c(1:7)]
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

Now our tidy training dataset contains 19622 observations of 53 variables and our tidy testing dataset contains 20 observations of 53 variables.

Model Building and Evaluation

We will be building two models: Classification Tree and Random Forest. We will compare the prediction accuracy of each models and the most accurate one will be used to predict the ‘classe’ of the 20 observations in the testing set.

Before we proceed with building our models, we need to load the following packages.

```
library(caret); library(rpart); library(rattle); library(rpart.plot); library(e1071);
library(randomForest)
```

To save computing time, 5 fold cross validation is considered for both models.

```
control <- trainControl(method = "cv", number = 5)
```

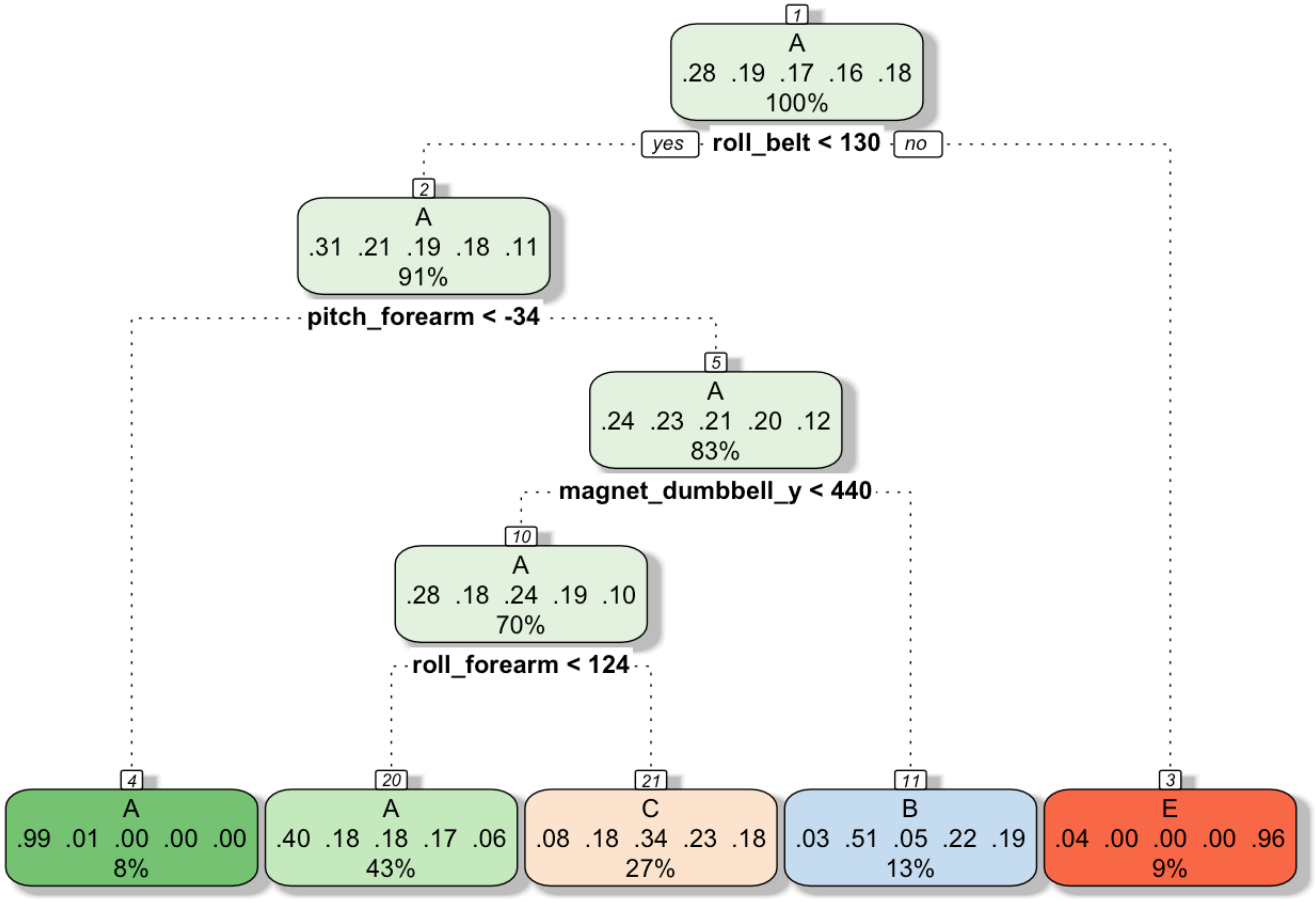
Data Splitting

The training data is splitted in to ‘myTraining’ and ‘myTesting’ datasets for model building and evaluation respectively. myTraining which constitute 60% of the dataset is used in developing our models and myTesting which consttute 40% of the dataset is used for predictive model assessment.

```
set.seed(4321)
inTrain<- createDataPartition(training$classe, p=.6)[[1]]
myTraining = training[inTrain,]
myTesting = training[-inTrain,]
```

Classification Tree

```
modfit_rpart<- train(classe ~.,data=myTraining, method="rpart", trControl = control)
fancyRpartPlot(modfit_rpart$finalModel)
```



Rattle 2016-Nov-02 19:40:47 tiblezogbagabriel

To evaluate the performance of the model and estimate the out-of-sample error, we predict the outcome ‘classe’ on the validation dataset (myTesting).

```
predict_rpart<- predict(modfit_rpart, myTesting)
confusionMatrix(predict_rpart, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2008  641  664  560  205
##          B   28  512   35  241  175
##          C  161  365  669  485  385
##          D    0    0    0    0    0
##          E   35    0    0    0  677
##
## Overall Statistics
##
##                Accuracy : 0.4927
##                  95% CI : (0.4816, 0.5039)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.337
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8996  0.33729  0.48904   0.0000  0.46949
## Specificity            0.6313  0.92430  0.78450   1.0000  0.99453
## Pos Pred Value         0.4924  0.51665  0.32397      NaN  0.95084
## Neg Pred Value         0.9406  0.85325  0.87909   0.8361  0.89277
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2559  0.06526  0.08527   0.0000  0.08629
## Detection Prevalence   0.5198  0.12631  0.26319   0.0000  0.09075
## Balanced Accuracy      0.7655  0.63080  0.63677   0.5000  0.73201
```

```
out_of_sample_Error_rpart<- 1 - as.numeric(confusionMatrix(predict_rpart, myTesting$c
lasse)$overall[1])
out_of_sample_Error_rpart
```

```
## [1] 0.5072648
```

From the confusion matrix, we can see that the accuracy is 0.493 and the out of sample error is 0.507. This indicates that classification tree does not predict the outcome classe very well.

Random Forest

```
modfit_rf<- randomForest(classe ~ ., data=myTraining, trControl = control)
print(modfit_rf)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = myTraining, trControl = control)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    2    1    1    0 0.001194743
## B   10 2259   10    0    0 0.008775779
## C    0   21 2030    3    0 0.011684518
## D    0    0   22 1906    2 0.012435233
## E    0    0    3    8 2154 0.005080831
```

To evaluate the performance of the model and estimate the out-of-sample error, we predict the outcome ‘classe’ on the validation dataset (myTesting).

```
predict_rf<- predict(modfit_rf, myTesting)
confusionMatrix(predict_rf, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232   14    0    0    0
##          B    0 1500    5    0    0
##          C    0    4 1360   14    0
##          D    0    0    3 1272    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.9926, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9881   0.9942   0.9891   0.9979
## Specificity            0.9975   0.9992   0.9972   0.9991   1.0000
## Pos Pred Value         0.9938   0.9967   0.9869   0.9953   1.0000
## Neg Pred Value         1.0000   0.9972   0.9988   0.9979   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1912   0.1733   0.1621   0.1834
## Detection Prevalence   0.2863   0.1918   0.1756   0.1629   0.1834
## Balanced Accuracy      0.9988   0.9937   0.9957   0.9941   0.9990
```

```
out_of_sample_Error_rf<- 1 - as.numeric(confusionMatrix(predict_rf, myTesting$classe)
$overall[1])
out_of_sample_Error_rf
```

```
## [1] 0.0054805
```

From the confusion matrix, we can see that the accuracy is 0.995 and the out of sample error is 0.005. This indicates that random forest can predict the outcome classe very well.

Prediction on Testing Dataset

With 99.5% accuracy, the random forest model is used to predict the 20 different test cases in the testing dataset.

```
pred <- predict(modfit_rf, newdata=testing)
PredictionResults <- data.frame(
  problem_id=testing$problem_id,
  predicted=pred)
print(PredictionResults)
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Conclusion

Random forest yielded much better results in comparison to classification tree. From this outcome, we learn that random forests improve predictive accuracy by generating a large number of bootstrapped trees based on random samples of variables, classifying a case using each tree in this new “forest”, and deciding a final predicted outcome by combining the results across all of the trees.

References

Human Activity Recognition http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)
Quick-R http://www.statmethods.net/ (http://www.statmethods.net/)