

Approach:

Our outcome variable is classe, a factor variable. For this data set, “participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions: - exactly according to the specification (Class A) - throwing the elbows to the front (Class B) - lifting the dumbbell only halfway (Class C) - lowering the dumbbell only halfway (Class D) - throwing the hips to the front (Class E)

Two models will be tested using decision tree and random forest. The model with the highest accuracy will be chosen as our final model.

Cross-validation

Cross-validation will be performed by subsampling our training data set randomly without replacement into 2 subsamples: TrainTrainingSet data (75% of the original Training data set) and TestTrainingSet data (25%). Our models will be fitted on the TrainTrainingSet data set, and tested on the TestTrainingSet data. Once the most accurate model is chosen, it will be tested on the original Testing data set.

Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the TestTrainingSet data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Our outcome variable “classe” is a factor variable. We split the Training dataset into TrainTrainingSet and TestTrainingSet datasets.

Install packages and load the required libraries

```
# install.packages("caret"); install.packages("randomForest"); install.packages("rpart") ;  
  
library(lattice); library(ggplot2); library(caret); library(randomForest); library(rpart); library(rpart.plot);  
  
## Warning: package 'lattice' was built under R version 3.1.1  
  
## Warning: package 'ggplot2' was built under R version 3.1.1  
  
## Warning: package 'caret' was built under R version 3.1.2  
  
## Warning: package 'randomForest' was built under R version 3.1.2  
  
## randomForest 4.6-10  
  
## Type rfNews() to see new features/changes/bug fixes.  
  
## Warning: package 'rpart.plot' was built under R version 3.1.2  
  
set.seed(1234)
```

```

# data load and clean up

trainingset <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
testingset <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!", ""))

# Perform exploratory analysis -

# dim(trainingset); dim(testingset); summary(trainingset); summary(testingset); str(tr
ainingset); str(testingset); head(trainingset); head(testingset);

# Delete columns with all missing values

trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]

# Delete variables are irrelevant to our current project: user_name, raw_timestamp_par
t_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and num_window (columns 1 to 7
).

trainingset <-trainingset[,-c(1:7)]
testingset <-testingset[,-c(1:7)]

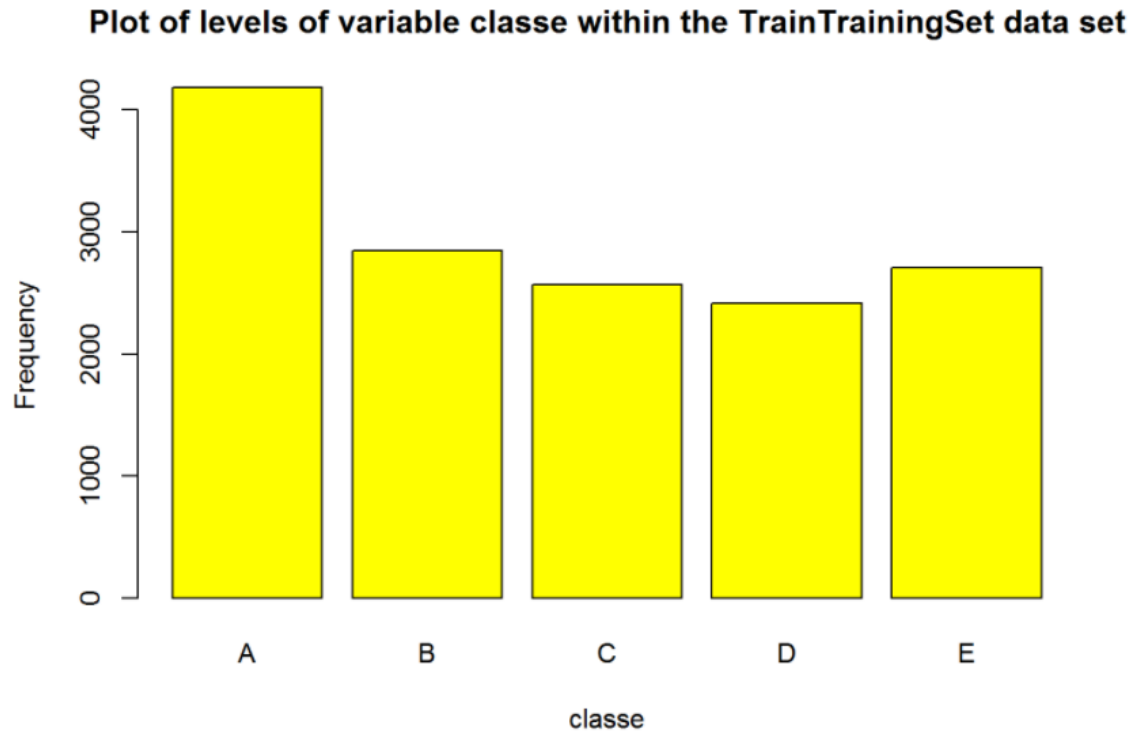
# partition the data so that 75% of the training dataset into training and the remaini
ng 25% to testing

traintrainset <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
TrainTrainingSet <- trainingset[traintrainset, ]
TestTrainingSet <- trainingset[-traintrainset, ]

# The variable "classe" contains 5 levels: A, B, C, D and E. A plot of the outcome var
iable will allow us to see the frequency of each levels in the TrainTrainingSet data s
et and # compare one another.

plot(TrainTrainingSet$classe, col="yellow", main="Plot of levels of variable classe wi
thin the TrainTrainingSet data set", xlab="classe", ylab="Frequency")

```



Based on the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent while level D is the least frequent.

Prediction model 1: Decision Tree

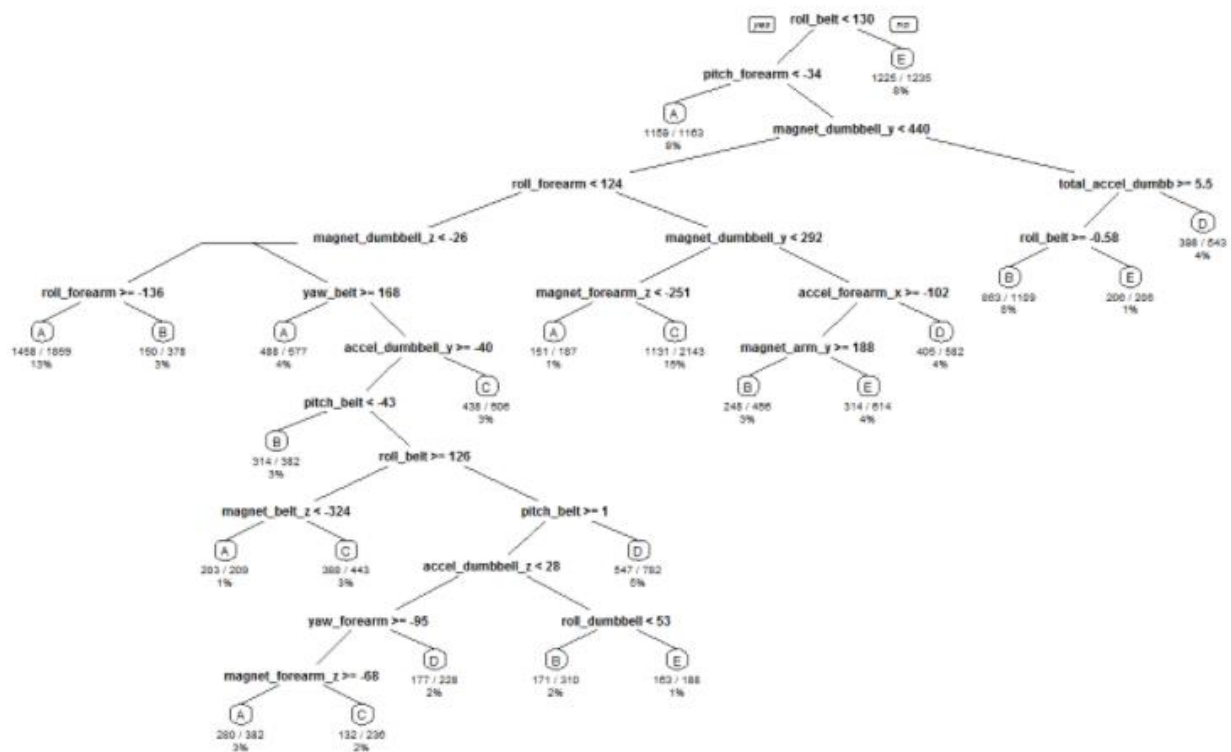
```
modell <- rpart(classe ~ ., data=TrainTrainingSet, method="class")

prediction1 <- predict(modell, TestTrainingSet, type = "class")

# Plot the Decision Tree

rpart.plot(modell, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```

Classification Tree



```
# Test results on our TestTrainingSet data set:
```

```
confusionMatrix(prediction1, TestTrainingSet$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    A    B    C    D    E
```

```
##           A 1235  157   16   50   20
```

```
##           B   55  568   73   80  102
```

```
##           C   44  125  690  118  116
```

```
##           D    41    64    50   508    38
##           E    20    35    26    48   625
##
## Overall Statistics
##
##           Accuracy : 0.739
##           95% CI : (0.727, 0.752)
##           No Information Rate : 0.284
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.67
##           McNemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity           0.885    0.599    0.807    0.632    0.694
## Specificity           0.931    0.922    0.900    0.953    0.968
## Pos Pred Value        0.836    0.647    0.631    0.725    0.829
## Neg Pred Value        0.953    0.905    0.957    0.930    0.933
## Prevalence            0.284    0.194    0.174    0.164    0.184
## Detection Rate        0.252    0.116    0.141    0.104    0.127
## Detection Prevalence  0.301    0.179    0.223    0.143    0.154
## Balanced Accuracy      0.908    0.760    0.854    0.792    0.831
```

Prediction model 2: Random Forest

```
model2 <- randomForest(classe ~. , data=TrainTrainingSet, method="class")

# Predicting:
```

```

prediction2 <- predict(model2, TestTrainingSet, type = "class")

# Test results on TestTrainingSet data set:
confusionMatrix(prediction2, TestTrainingSet$classe)

## Confusion Matrix and Statistics

##
##           Reference
## Prediction    A    B    C    D    E
##
##      A 1394     3     0     0     0
##      B   1  944    10     0     0
##      C   0   2  843     6     0
##      D   0   0   2  798     0
##      E   0   0   0   0  901
##
## 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   0.995   0.986   0.993   1.000
## Specificity           0.999   0.997   0.998   1.000   1.000

```

## Pos Pred Value	0.998	0.988	0.991	0.997	1.000
## Neg Pred Value	1.000	0.999	0.997	0.999	1.000
## Prevalence	0.284	0.194	0.174	0.164	0.184
## Detection Rate	0.284	0.192	0.172	0.163	0.184
## Detection Prevalence	0.285	0.195	0.174	0.163	0.184
## Balanced Accuracy	0.999	0.996	0.992	0.996	1.000

Decision on which Prediction Model to Use:

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to Decision Tree model with 0.739 (95% CI: (0.727, 0.752)). The Random Forests model is chosen. The expected out-of-sample error is estimated at 0.005, or 0.5%.

Submission

Here is the final outcome based on the Prediction Model 2 (Random Forest) applied against the Testing dataset

```
# predict outcome levels on the original Testing data set using Random Forest algorithm
m

predictfinal <- predict(model2, testingset, type="class")

predictfinal

##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

##  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
```