

Prediction Assignment

Neil Fernandes

4/7/2017

Approach

The outcome variable is `classe`, a factor variable with 5 levels. The levels are: 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)

Predictions in this report are based on optimal accuracy and out of sample error. Two models will be tested using decision tree and random forest algorithms.

Cross Validation

Cross-validation will be performed by subsampling our training data set randomly without replacement (With replacement is bootstrapping) into 2 subsamples: `subTraining` data (70% of the original Training data set) and `subTesting` data (30%, which is not trained on in an effort to avoid overfitting). Our models will be fitted on the `subTraining` data set, and then tested on the `subTesting` data. Once the most accurate model is found, it will be tested on the original Testing data set.

Expected Error rate

The expected out-of-sample error will correspond to the quantity: $1 - \text{accuracy}$ in the cross-validation data. The expected out of sample error will correlate to the expected missclassification observations/total observations in the Test data set which is equal to $1 - \text{cross validation accuracy}$.

Notes

Features with all missing values will be discarded as well as features that are irrelevant. All other features will be kept as relevant variables. Decision tree and random forest algorithms are known for their ability to detect the features that are important for classification. Feature selection is built in, so I am skipping feature selection because it is already chosen.

As for the limitation in this study, the observation data used in the analyses was collected from 6 young health participants in an experiment using a Microsoft Kinect. This leads us to believe the model should be at least 95% accurate. This model is based on 6 young healthy participants so any deviations from those types of participants may lead to less accurate results.

Results

Install the necessary packages and libraries

```
#install.packages(caret); #only need to install packages once and then just call on libraries
#install.packages(randomForest);
#install.packages(rpart);
#install.packages(rpart.plot)
#install.packages('e1071', dependencies=TRUE) #Need this package as well
library(lattice);
library(ggplot2);
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(caret);
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
library(randomForest);
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
```

```
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
```

```
##
```

```
##      margin
```

```
library(rpart);
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
set.seed(385) #For Reproducibility (If you want to get the same results)
```

```
training <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

```
testing <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

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

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

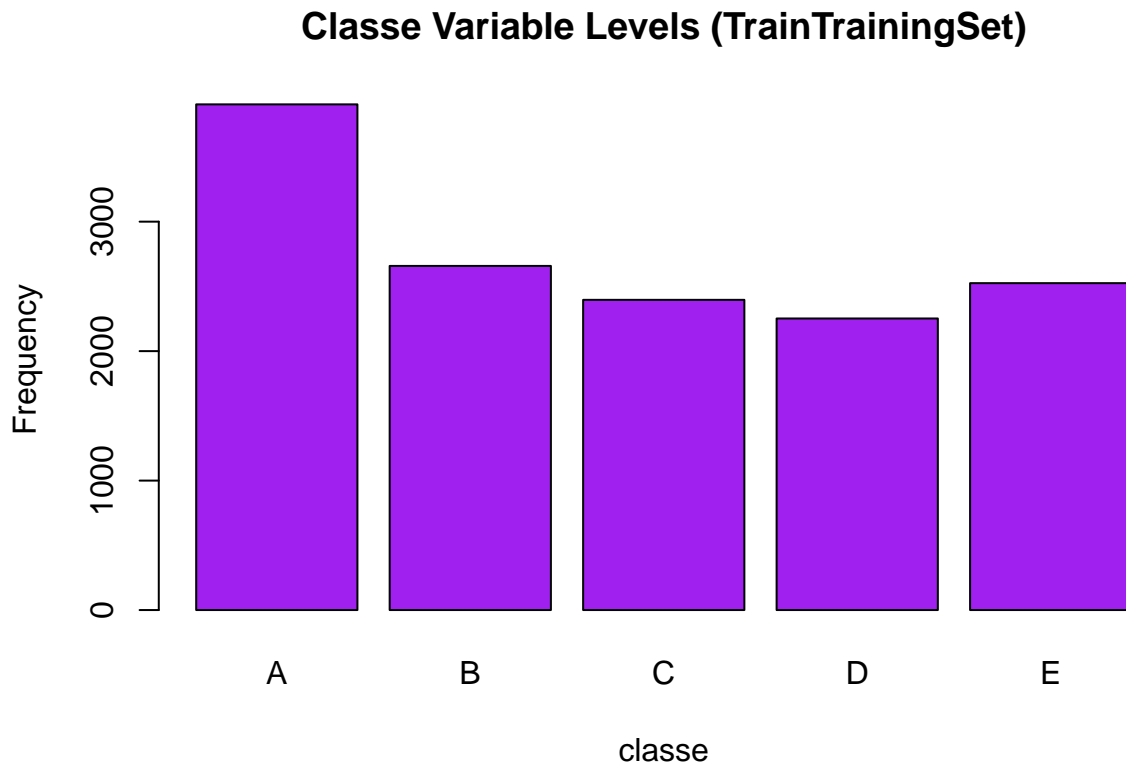
```
# Perform exploratory analysis, -  
# dim(trainingset); dim(testingset);  
#summary(trainingset); summary(testingset);  
#str(trainingset); str(testingset);  
#head(trainingset); head(testingset);  
#these are commented out to save space
```

```
# Delete columns with all missing values, we do not these to offset out results  
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_part_1, raw_timestamp_part_2)  
#We really just want to know about their health and times  
trainingset <-trainingset[,-c(1:7)]  
testingset <-testingset[,-c(1:7)]
```

```
# partition the data so that 70% of the training dataset into training and the remaining 30% to testing
traintrainset <- createDataPartition(y=trainingset$classe, p=0.70, list=FALSE)
TrainTrainingSet <- trainingset[traintrainset, ]
TestTrainingSet <- trainingset[-traintrainset, ]

# A plot of the outcome variable will allow us to see the frequency of each levels in the TrainTrainingSet
plot(TrainTrainingSet$classe, col="purple", main="Classe Variable Levels (TrainTrainingSet)", xlab="classe")
```



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 1668    9    0    0    0
##           B    6 1127    2    0    0
##           C    0    3 1023    5    0
```

```
##           D      0      0      1  959      2
##           E      0      0      0      0 1080
##
## Overall Statistics
##
##           Accuracy : 0.9952
##           95% CI : (0.9931, 0.9968)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.994
## Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9964  0.9895  0.9971  0.9948  0.9982
## Specificity      0.9979  0.9983  0.9984  0.9994  1.0000
## Pos Pred Value   0.9946  0.9930  0.9922  0.9969  1.0000
## Neg Pred Value   0.9986  0.9975  0.9994  0.9990  0.9996
## Prevalence       0.2845  0.1935  0.1743  0.1638  0.1839
## Detection Rate   0.2834  0.1915  0.1738  0.1630  0.1835
## Detection Prevalence 0.2850  0.1929  0.1752  0.1635  0.1835
## Balanced Accuracy 0.9971  0.9939  0.9977  0.9971  0.9991
```

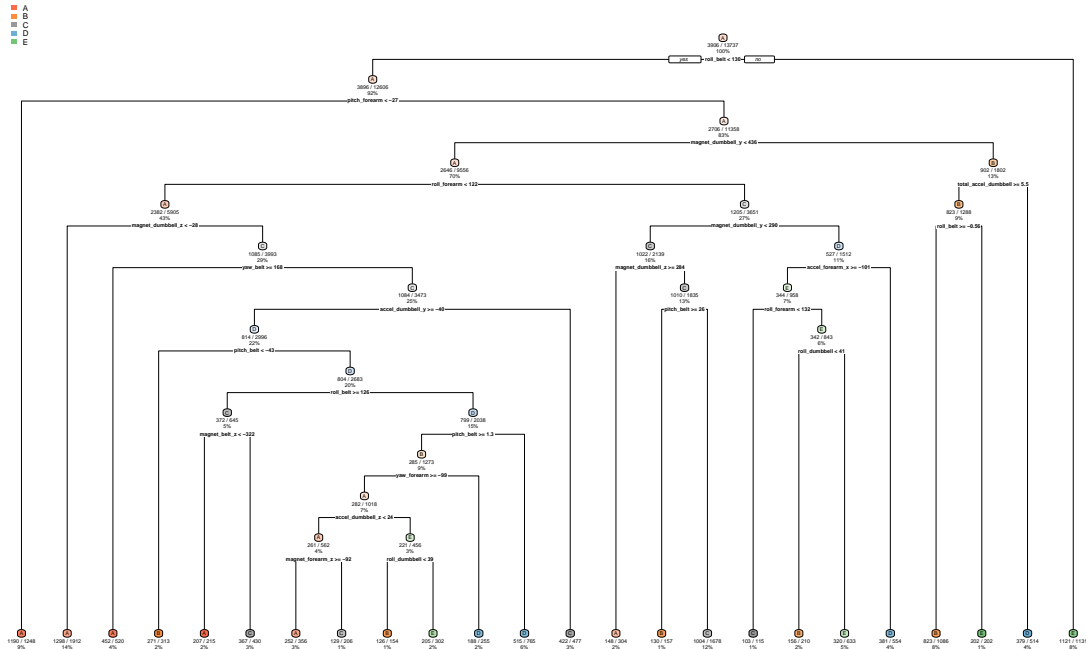
Decision Tree

```
model1 <- rpart(classe ~ ., data=TrainTrainingSet, method="class")

prediction1 <- predict(model1, TestTrainingSet, type = "class")

# Plot the Decision Tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```

Classification Tree



```
# Testing results on TestTrainingSet data set:
confusionMatrix(prediction1, TestTrainingSet$classe)
```

Confusion Matrix and Statistics

```
##
##           Reference
## Prediction   A    B    C    D    E
##           A 1501  268   22  126   56
##           B   53  617   48   16   64
##           C   34  117  863  146  128
##           D   61   75   71  602   56
##           E   25   62   22   74  778
```

Overall Statistics

```
##
##           Accuracy : 0.741
##           95% CI : (0.7296, 0.7522)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##           Kappa : 0.6706
##           McNemar's Test P-Value : < 2.2e-16
```

Statistics by Class:

```
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.8967   0.5417   0.8411   0.6245   0.7190
## Specificity      0.8879   0.9619   0.9125   0.9466   0.9619
## Pos Pred Value   0.7608   0.7732   0.6700   0.6960   0.8096
## Neg Pred Value   0.9558   0.8974   0.9645   0.9279   0.9383
```

## Prevalence	0.2845	0.1935	0.1743	0.1638	0.1839
## Detection Rate	0.2551	0.1048	0.1466	0.1023	0.1322
## Detection Prevalence	0.3353	0.1356	0.2189	0.1470	0.1633
## Balanced Accuracy	0.8923	0.7518	0.8768	0.7855	0.8405

Decision

Random Forest seems to be a better choice than Decision Trees. Random Forest was .9954 accurate with a 95% Confidence Interval (0.9933, 0.997) while Decision Tree was .741 with a 95% Confidence Interval (0.7296, 0.7522). The expected out of sample error is .5%.

Submission

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