Activity Prediction Using Groupware Data

Introduction

One of the interestng things is to gather data about individuals' excercise and analyze the manner in which they did the exercise. Using modern gadgets such as Jawbone Up, Nike FuelBand, and Fitbit facilitates collecting large amounts of data about personal activity. In this report we will data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal to predict the manner in which they did the exercise.

Dataset

The data for this project is provided by Groupware. Training set is available Here and Test set is available Here. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways, while placing accelerometers on the belt, forearm, arm, and dumbell

Data Exploration And Cleaning

Loading Libraries

```
#load libraries
library(caret)
library(randomForest)
library(rpart)
library(dplyr)
library(rattle)
```

Exploring Data

```
traind<-read.csv("./data/pml-training.csv",
              stringsAsFactors = FALSE,
              header = TRUE,
              na.strings = c("","NA"))

testd<-read.csv("./data/pml-testing.csv",
                stringsAsFactors = FALSE,
                header = TRUE,
                na.strings = c("","NA"))

dim(traind)
```

```
## [1] 19622   160
```

We can see that we have 160 column, including the **classe** variable which we want to predict in the test set.

Cleaning Data

Here we will: - remove any columns that include more than 80% Nas.

```
#check percentage of NAs
check_na<-sapply(1:dim(traind)[2], function(x) mean(is.na(traind[,x])))

#get columns with less than 20% Na
traind<- traind[,which(check_na<0.2)]
testd<- testd[,which(check_na<0.2)]
```

- remove columns that include ids and data that do not come from measurements, and won't help in our predictions.

```
#remove the first 7 columns as they seem to have no effect (id, name..etc.)
traind<-traind[,8:dim(traind)[2]]
testd<-testd[,8:dim(testd)[2]]
```

- convert the **classe** variable into factor as it is our output variable.

```
traind$classe<-as.factor(traind$classe)
```

Prediction

Data Partitioning

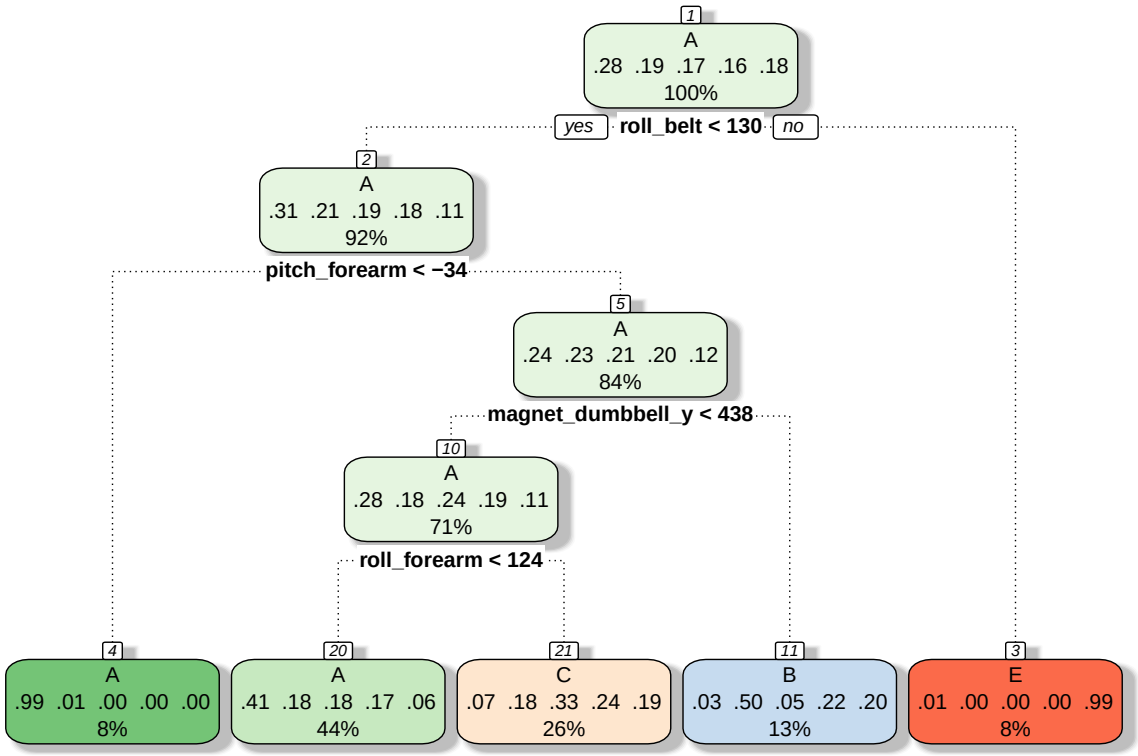
Here we'll devide the training set into 2 parts 60%, 40%

```
##divide training set into 2 parts 60%, 40%
inTrain<-createDataPartition(y=traind$classe,p=0.6,list = F)

training <- traind[inTrain,]
testing <- traind[-inTrain,]
```

Decision Tree

Here we will start with a basic classification tree

```
model<-train(classe ~ .,data = training, method = "rpart")

fancyRpartPlot(model$finalModel, sub="")
```



If we check confusion matrix, we can see that the result is unsatisfactory, the accuracy is low. So we'll try a random forest.

```
confusionMatrix(testing$classe, predict(model, testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2025   36  168    0    3
##          B  638  513  367    0    0
##          C  631   33  704    0    0
##          D  587  237  462    0    0
##          E  207  180  386    0  669
##
## Overall Statistics
##
##                Accuracy : 0.4985
##                  95% CI : (0.4873, 0.5096)
##     No Information Rate : 0.521
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3444
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4954  0.51351  0.33733       NA  0.99554
## Specificity            0.9449  0.85322  0.88470   0.8361  0.89225
## Pos Pred Value         0.9073  0.33794  0.51462       NA  0.46394
## Neg Pred Value         0.6325  0.92320  0.78651       NA  0.99953
## Prevalence             0.5210  0.12733  0.26600   0.0000  0.08565
## Detection Rate         0.2581  0.06538  0.08973   0.0000  0.08527
## Detection Prevalence   0.2845  0.19347  0.17436   0.1639  0.18379
## Balanced Accuracy      0.7201  0.68337  0.61101       NA  0.94389
```

Random Forest

Here we will fit a random forest and check the accuracy.

```
model2<- randomForest(classe ~ ., data = training)

predict_class<-predict(model2, testing, type = "class")
```

Let's look at the confusion matrix

```
confusionMatrix(testing$classe,predict_class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    1    0
##          B   14 1501    3    0    0
##          C    0    6 1362    0    0
##          D    0    0   17 1268    1
##          E    0    0    2    4 1436
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9919, 0.9955)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9938   0.9960   0.9841   0.9961   0.9993
## Specificity            0.9998   0.9973   0.9991   0.9973   0.9991
## Pos Pred Value         0.9996   0.9888   0.9956   0.9860   0.9958
## Neg Pred Value         0.9975   0.9991   0.9966   0.9992   0.9998
## Prevalence             0.2861   0.1921   0.1764   0.1622   0.1832
## Detection Rate         0.2843   0.1913   0.1736   0.1616   0.1830
```

```
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9968   0.9967   0.9916   0.9967   0.9992
```

It seems that the random forest gave a good level of accuracy that we can rely on. So we'll use our test set to predict **classe**.

```
p<-predict(model2, testd, type = "class")
```

And here is the predicition for the 20 test cases:

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