Prediction Assignment

Arkhipenko Anastasia

The Task

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

1.Loading Packages

2.Loading Data

Let N be the total length of your first and last names. Also, choose a random number a [0.4, 0.6] and round it to one decimal place.

```
set.seed(42) #for reproducibility

Train_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

Train <- read.csv(url(Train_URL), na.strings=c("NA","#DIV/0!",""))
Test <- read.csv(url(Test_URL), na.strings=c("NA","#DIV/0!",""))

#Data partioning
inTrain <- createDataPartition(y=Train$classe, p=0.75, list=FALSE)
Train_1 <- Train[inTrain, ]
Test_1 <- Train[-inTrain, ]
dim(Train_1)
```

```
## [1] 14718   160
```

```
dim(Test_1)
```

```
## [1] 4904  160
```

```
dim(Test)
```

```
## [1]  20 160
```

3.Data Cleaning

```
#Removing near zero variance variables
nzv <- nearZeroVar(Train_1, saveMetrics=TRUE)
Train_1 <- Train_1[,nzv$nzv==FALSE]

nzv<- nearZeroVar(Test_1,saveMetrics=TRUE)
Test_1 <- Test_1[,nzv$nzv==FALSE]
```

```
Train_1 <- Train_1[c(-1)]

#Removing variables with 70% and more NAs
training <- Train_1
for(i in 1:length(Train_1)) {
    if( sum( is.na(Train_1[, i] ) ) /nrow(Train_1) >= .7) {
        for(j in 1:length(training)) {
            if(length(grep(names(Train_1[i]), names(training)[j]) ) == 1)  {
                training <- training[ , -j]
            }
        }
    }
}

# Setting back to the original name and removing etra variable
Train_1 <- training
rm(training)

#Transforming the Test_1 and Test data sets to make them have the same features
a <- colnames(Train_1)
b <- colnames(Train_1[, -58]) #removing "classe" column
Test_1 <- Test_1[a] #allow only variables in Test_1 that are also in Train_1
Test <- Test[b] #allow only variables in Test that are also in Train_1

dim(Train_1)
```

```
## [1] 14718    58
```

```
dim(Test_1)
```

```
## [1] 4904   58
```

```
dim(Test)
```

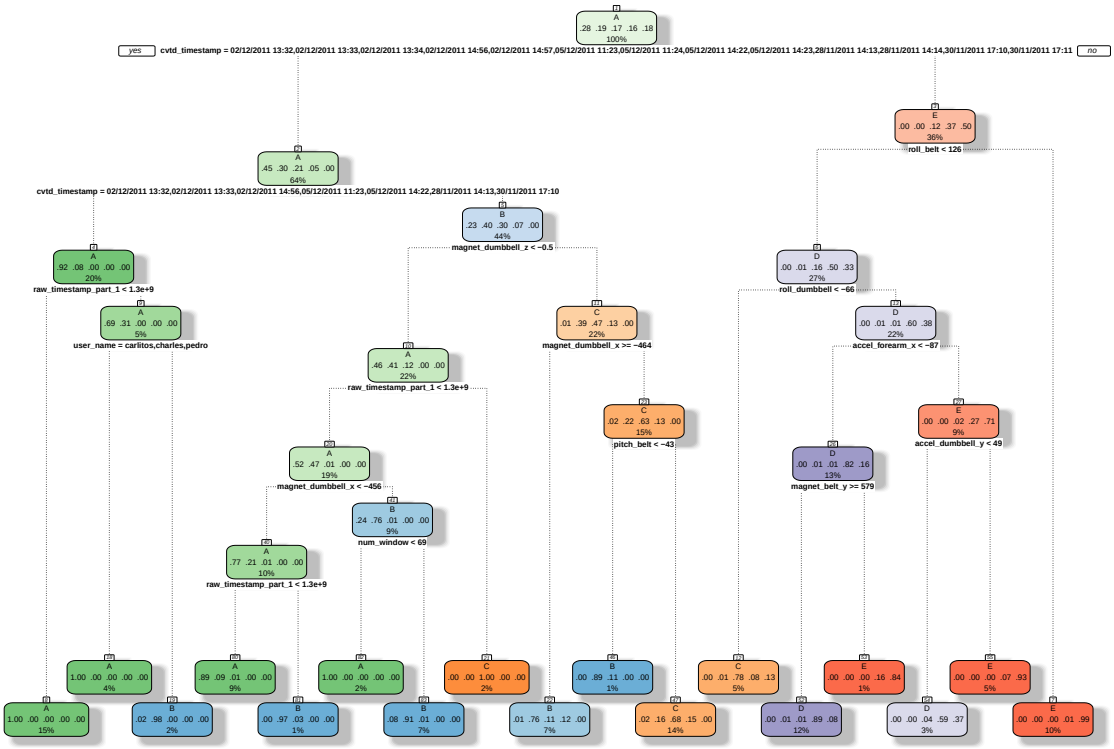
```
## [1] 20 57
```

```
for (i in 1:length(Test)) {
    for(j in 1:length(Train_1)) {
        if(length(grep(names(Train_1[i]), names(Test)[j]) ) == 1)  {
            class(Test[j]) <- class(Train_1[i])
        }
    }
}

#Getting the same classes between Test and Train_1
Test <- rbind(Train_1[2, -58], Test)
Test <- Test[-1,]
```

4.Building Prediction Models

Decision Trees. It's fast and simple so lets use it first.

```
set.seed(42)

model_1 <- rpart(classe ~ ., data = Train_1, method = "class")
fancyRpartPlot(model_1)
```



Rattle 2018−Nov−25 11:55:07 Anastasia

```
predictions_1 <- predict(model_1, Test_1[, 1:58], type = "class")
confusion_matrix_1 <- confusionMatrix(predictions_1, Test_1$classe)
confusion_matrix_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1335   37    4    1    0
##          B   47  793   58   34    0
##          C   13  114  779  137   43
##          D    0    5   14  601  112
##          E    0    0    0   31  746
##
## Overall Statistics
##
##                Accuracy : 0.8675
##                  95% CI : (0.8576, 0.8768)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8325
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9570   0.8356   0.9111   0.7475   0.8280
```

```
## Specificity            0.9880   0.9649   0.9242   0.9680   0.9923
## Pos Pred Value         0.9695   0.8509   0.7173   0.8210   0.9601
## Neg Pred Value         0.9830   0.9607   0.9801   0.9513   0.9624
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2722   0.1617   0.1588   0.1226   0.1521
## Detection Prevalence   0.2808   0.1900   0.2215   0.1493   0.1584
## Balanced Accuracy      0.9725   0.9002   0.9176   0.8578   0.9101
```

Random Forests

```
model_2 <- randomForest(classe ~ ., data = Train_1)
predictions_2 <- predict(model_2, Test_1[, 1:58], type = "class")
confusion_matrix_2 <- confusionMatrix(predictions_2, Test_1$classe)
confusion_matrix_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    1    0    0
##          C    0    0  853    5    0
##          D    0    0    1  799    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9971, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9977   0.9938   1.0000
## Specificity            1.0000   0.9997   0.9988   0.9998   1.0000
## Pos Pred Value         1.0000   0.9989   0.9942   0.9988   1.0000
## Neg Pred Value         1.0000   1.0000   0.9995   0.9988   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1739   0.1629   0.1837
## Detection Prevalence   0.2845   0.1937   0.1750   0.1631   0.1837
## Balanced Accuracy      1.0000   0.9999   0.9982   0.9968   1.0000
```

5.Predictions on the Test Data

Finally, we have to predict the 20 test cases for the quiz. Random Forest shows the highest accuracy between these two models I used for training. So lets use it for final testing:

```
final_predictions <- predict(model_2, Test, type = "class")
final_predictions
```

```
##  1 21  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
```

Conclusion. The Decision tree and Random Forest models which were built in this project are very likely to overfit the data. To improve performance (in particular their generalization ability we could, for example, remove the "user name"" and "time reference" features. And the dataset can be improved by collecting information about height, weight and other physical characteristics of the user.