Project

Background

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project 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)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

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

Remove irrelevant data

```
training<- training[ , colSums(is.na(training)) == 0]
remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'c
vtd_timestamp', 'new_window', 'num_window')
training<- training[, -which(names(training) %in% remove)]
```

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
zeroVar= nearZeroVar(training[sapply(training, is.numeric)], saveMetrics = TRU
E)
trainingNonZero= training[,zeroVar[, 'nzv']==0]
corr <- cor(na.omit(trainingNonZero[sapply(trainingNonZero, is.numeric)]))
dim(corr)
```

```
## [1] 52 52
```

Remove correlations above 85 percent.

```
corrMatrix <- cor(na.omit(trainingNonZero[sapply(trainingNonZero, is.numeri
c)]))
corr <- expand.grid(row = 1:52, col = 1:52)
corr$correlation <- as.vector(corrMatrix)

removecor = findCorrelation(corrMatrix, cutoff = .85, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 2  and column  11 with corr  0.884
##   Means:  0.228 vs 0.154 so flagging column 2
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.09 vs 0.154 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.158 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.082 vs 0.161 so flagging column 33
## All correlations <= 0.85
```

```
trainingWithoutCorrelated = trainingNonZero[,-removecor]

dim (trainingWithoutCorrelated)
```

```
## [1] 19622    45
```

Finalize Data Set

```
inTrain <- createDataPartition(y=trainingWithoutCorrelated$classe, p=0.7, list=
FALSE)
training <- trainingWithoutCorrelated[inTrain,]; testing <- trainingWithoutCorr
elated[-inTrain,]
dim(training)
```

```
## [1] 13737    45
```

```
dim(testing)
```

```
## [1] 5885   45
```

Try Regression Tree

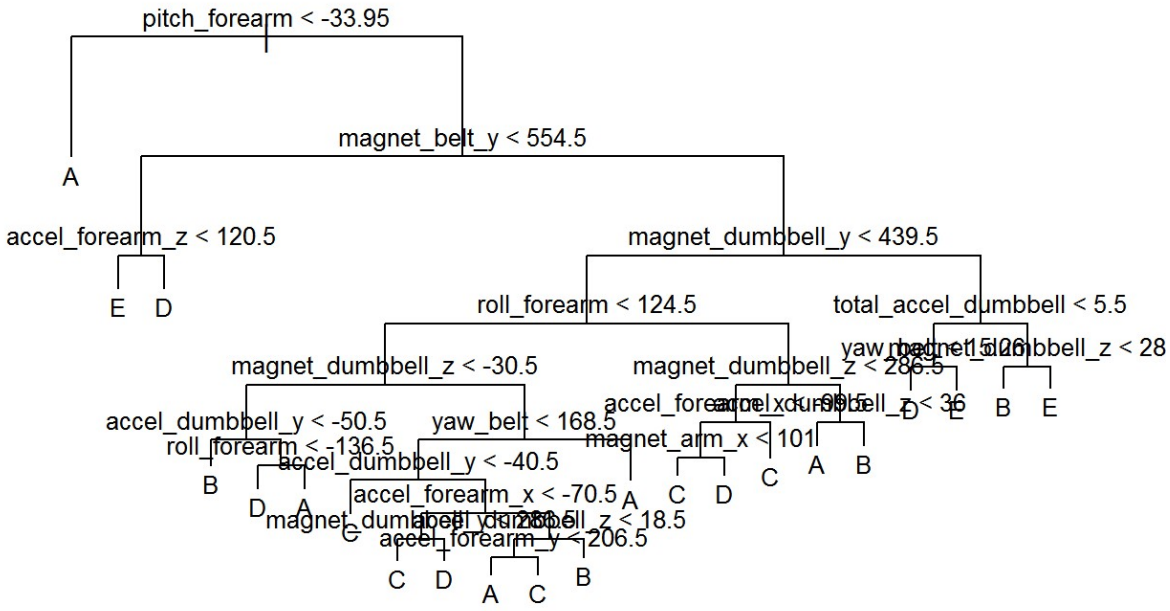
```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.2.2
```

```
set.seed(345)
treetraining=tree(classe~.,data=training)
summary(treetraining)
```

```
##
## Classification tree:
## tree(formula = classe ~ ., data = training)
## Variables actually used in tree construction:
##  [1] "pitch_forearm"        "magnet_belt_y"        "accel_forearm_z"
##  [4] "magnet_dumbbell_y"    "roll_forearm"         "magnet_dumbbell_z"
##  [7] "accel_dumbbell_y"     "yaw_belt"             "accel_forearm_x"
## [10] "accel_dumbbell_z"     "accel_forearm_y"      "magnet_arm_x"
## [13] "total_accel_dumbbell"
## Number of terminal nodes:  22
## Residual mean deviance:  1.643 = 22530 / 13720
## Misclassification error rate: 0.3253 = 4469 / 13737
```

```
plot(treetraining)
text(treetraining,pretty=0, cex =.8)
```




```
predictedtree=predict(treetraining,testing,type="class")
predictedMatrix = with(testing,table(predictedtree,classe))
sum(diag(predictedMatrix))/sum(as.vector(predictedMatrix)) # error rate
```

```
## [1] 0.6679694
```

Not the best result.Let's try to prune the tree.

```
cv.training=cv.tree(treetraining,FUN=prune.misclass)
cv.training
```

```
## $size
##  [1] 22 21 20 19 18 17 16 15 14 13 12  8  6  5  1
##
## $dev
##  [1] 4681 4684 4705 4721 4752 5092 5092 5517 5623 5623 6459 6548 6925 7223
## [15] 9831
##
## $k
##  [1]  -Inf   9.0  42.0  49.0  66.0 124.0 126.0 146.0 162.0 165.0 203.0
## [12] 213.0 245.5 305.0 655.5
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prunedtraining=prune.misclass(treetraining,best=18)

predictedtree=predict(prunedtraining,testing,type="class")
predMatrix = with(testing,table(predictedtree,classe))
sum(diag(predMatrix))/sum(as.vector(predMatrix)) # error rate
```

```
## [1] 0.6540357
```

Random Forests

```
require(randomForest)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
```

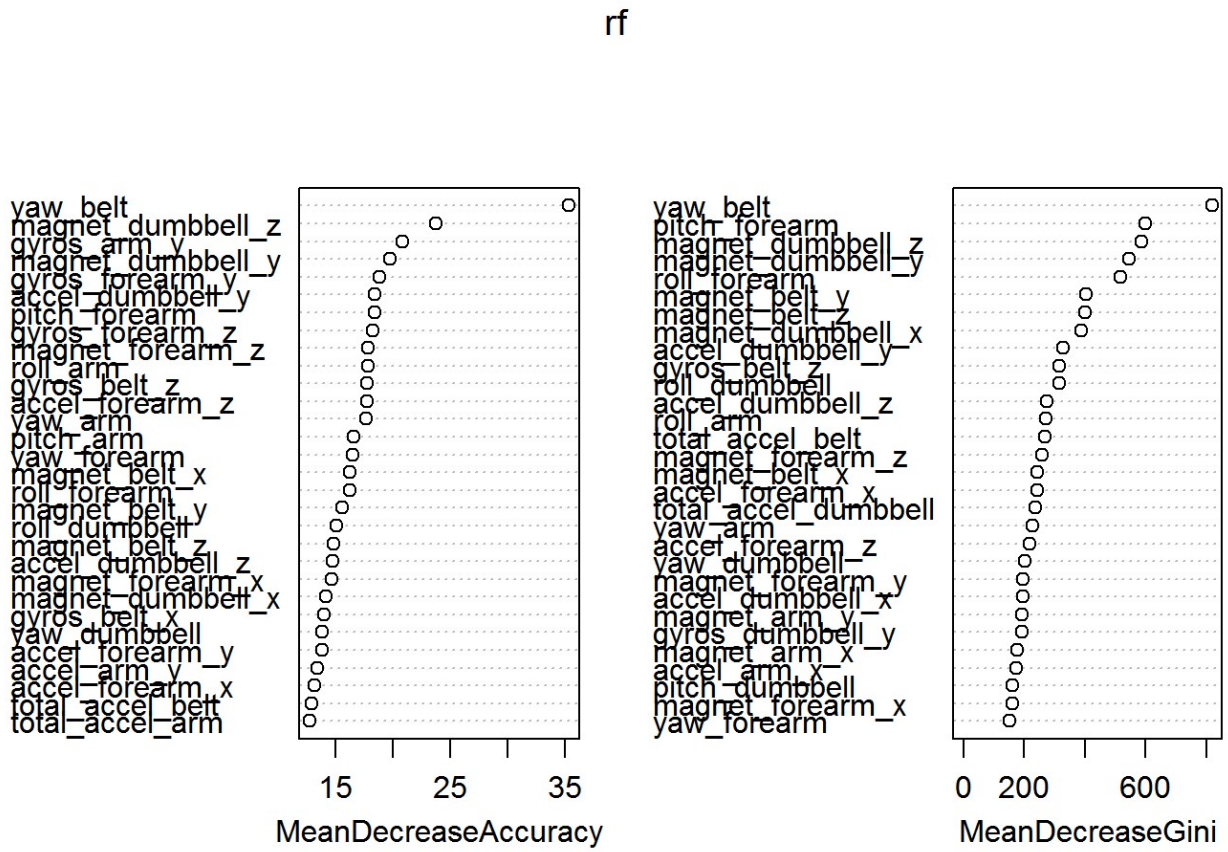
```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(345)

rf=randomForest(classe~.,data=training,ntree=100, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 100,      impor
tance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.92%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    3    1    1    1 0.001536098
## B   24 2622   11    1    0 0.013544018
## C    1   28 2365    2    0 0.012938230
## D    2    0   38 2209    3 0.019094139
## E    0    3    1    7 2514 0.004356436
```

```
varImpPlot(rf,)
```



```
predictedTree=predict(rf,testing,type="class")
predMatrix = with(testing,table(predictedTree,classe))
sum(diag(predMatrix))/sum(as.vector(predMatrix)) # error rate
```

```
## [1] 0.9913339
```

Looks like the best.

Let's test

Testing

```
testing= read.csv("pml-testing.csv", na.strings=c("", "NA", "NULL"))
answers <- predict(rf, testing)
answers
```

```
##  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
```