Project for Coursera Practical Machine Learning

Our 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. We start by reading and partitioning the data:

```
library(caret)
```

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

```
allTrainData<-read.csv("./pml-training.csv")
realTestData<-read.csv("./pml-testing.csv")
dim(allTrainData)
```

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

```
indices<-createDataPartition(y=allTrainData$classe, p=0.6, list=FALSE)
train<-allTrainData[indices,]
test<-allTrainData[-indices,]
```

We have now divided the Training data into 60% for Training, and 40% for testing. We now remove all of those columns that are not going to be useful for the predictions, for example, the names and the raw timestamps. Also, we are eliminating those that have NA

```
cols <- sapply(1:dim(train)[2], function(x) all(!is.na(train[,x]) & is.numeric(train[,x])))
cols[1:7]<-F
```

Now we use a random forest classifier to fit the data

```
library(randomForest)
```

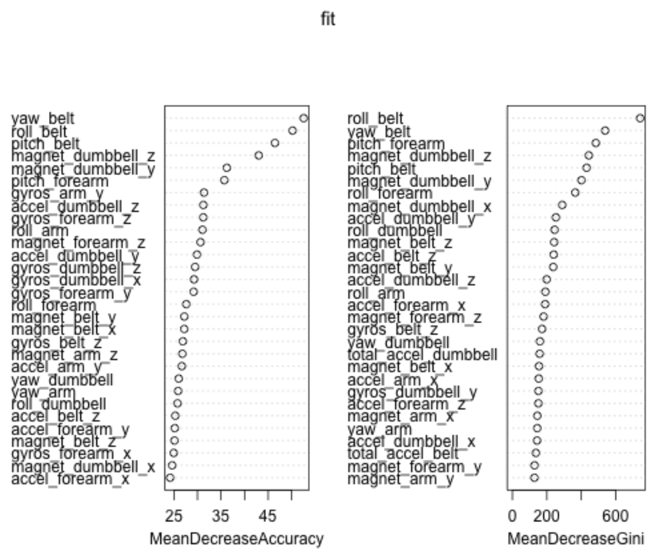
```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
mat<-predict(preProcess(train[,cols], method=c("center","scale")),train[,cols])
mat[,"classe"]<-train$classe
fit<-randomForest(classe ~ ., data=mat, importance=T, ntree=400, norm.votes=T)
fit
```

```
## 
## Call:
##  randomForest(formula = classe ~ ., data = mat, importance = T,      ntree = 400, norm.votes = T) 
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 7
## 
##         OOB estimate of  error rate: 0.63%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3341    6    0    0    1    0.002091
## B   10 2263    6    0    0    0.007021
## C    0   13 2039    2    0    0.007303
## D    0    0   26 1903    1    0.013990
## E    0    0    2    7 2156    0.004157
```

As you can see, using 400 trees and 7 variables at each split (the default) we get a good accuracy. The following graph allows us to see which are the most important features:

```
varImpPlot(fit)
```



We keep these predictors and train again:

```
new_cols<-
c("roll_belt","pitch_belt","yaw_belt","gyros_belt_z","accel_belt_z","magnet_belt_x","magnet_belt_y","magnet_

new_mat<-train[,new_cols]
new_mat[,"classe"]<-train[,"classe"]
new_fit<-randomForest(classe ~ ., n_tree=400, data = new_mat, importance=TRUE)
new_fit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = new_mat, n_tree = 400,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    3    1    0    1    0.001493
## B   13 2264    2    0    0    0.006582
## C    0   14 2033    7    0    0.010224
## D    0    0   18 1911    1    0.009845
## E    0    0    3    8 2154    0.005081
```

Our new fitted model shows improvement with respect to the previous one. Now we do a cross-validation. In this case, we will try a 15 fold cross validation (This part of the code might take some time to run)

```
#install.packages("cvTools")
library(cvTools)
```

```
## Loading required package: robustbase
```

```
foldings <- cvFolds(dim(new_mat)[1], K = 15, R = 1)
errors <- rep(NA,15)
for (i in 1:15) {
  ind <- foldings$subsets[foldings$which == i]
  temporal_model <- randomForest(classe ~ ., data=new_mat[-ind,])
  result <- predict(temporal_model,new_mat[ind,])
  errors[i] <- sum(result != new_mat[ind,"classe"]) / length(result)
}
errors
```

```
##  [1] 0.006361 0.002548 0.008917 0.010191 0.007643 0.006369 0.000000
##  [8] 0.010191 0.005096 0.003822 0.010191 0.007643 0.003822 0.011465
## [15] 0.007643
```

Now that the model is ready, we can use it with the test data.

```
test <- predict(preProcess(train[,cols],
method=c("center","scale")),train[,cols],realTestData[,cols])
test[,"classe"] <-realTestData[,cols]$classe
OurPrediction <- predict(new_fit, test)
```