Machine Learning 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 (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

The test data are available here:

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. 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.

Loading Data

```
trainingRAW <- read.csv('pml-training.csv', na.strings= c("NA",""))
testingRAW <- read.csv('pml-testing.csv', na.strings= c("NA",""))
```

Cleaning Data

Data set has a lot of columns with NA. We strip them and also remove some columns that clearly not needed for prediction (i.e. raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, etc.)

```
training <- trainingRAW[, (colSums(is.na(trainingRAW)) == 0)]
testing <- testingRAW[, (colSums(is.na(trainingRAW)) == 0)]

training <- training[, !grepl("timestamp|window|user_name|X",names(training))]
testing <- testing[, !grepl("timestamp|window|user_name|X",names(testing))]
```

Create Test vs. Validation Set

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.1
```

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

## Warning: package 'ggplot2' was built under R version 3.1.1
```

```
set.seed(12345)
inTrain = createDataPartition(y = training$classe, p = 0.7, list = FALSE)
trainData = training[inTrain, ]
validationData = training[-inTrain, ]
```

Preprocessing and Model Selection

As shown below, there's signicant # of variables correlated. Therefore, we use preprocess data with Principal Component Analysis (PCA). Random forest method is used since Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way.

```
M<-abs(cor(trainData[,-53]))
diag(M)<-0
which(M>0.8, arr.ind=T)
```

```
##                  row col
## yaw_belt           3   1
## total_accel_belt   4   1
## accel_belt_y       9   1
## accel_belt_z      10   1
## accel_belt_x       8   2
## magnet_belt_x     11   2
## roll_belt          1   3
## roll_belt          1   4
## accel_belt_y       9   4
## accel_belt_z      10   4
## pitch_belt         2   8
## magnet_belt_x     11   8
## roll_belt          1   9
## total_accel_belt   4   9
## accel_belt_z      10   9
## roll_belt          1  10
## total_accel_belt   4  10
## accel_belt_y       9  10
## pitch_belt         2  11
## accel_belt_x       8  11
## gyros_arm_y       19  18
## gyros_arm_x       18  19
## magnet_arm_x      24  21
## accel_arm_x       21  24
## magnet_arm_z      26  25
## magnet_arm_y      25  26
## accel_dumbbell_x  34  28
## accel_dumbbell_z  36  29
## gyros_dumbbell_z  33  31
## gyros_forearm_z   46  31
## gyros_dumbbell_x  31  33
```

```
## gyros_forearm_z   46  33
## pitch_dumbbell    28  34
## yaw_dumbbell      29  36
## gyros_forearm_z   46  45
## gyros_dumbbell_x  31  46
## gyros_dumbbell_z  33  46
## gyros_forearm_y   45  46
```

```
preProc <- preProcess(trainData[, -53], method = "pca")
trainPC <- predict(preProc, trainData[, -53])
testPC <- predict(preProc, testing[, -53])
validationPC <- predict(preProc, validationData[, -53])
modelFit <- train(trainData$classe ~ ., method = "rf", data = trainPC, trControl = trainControl(method =
```

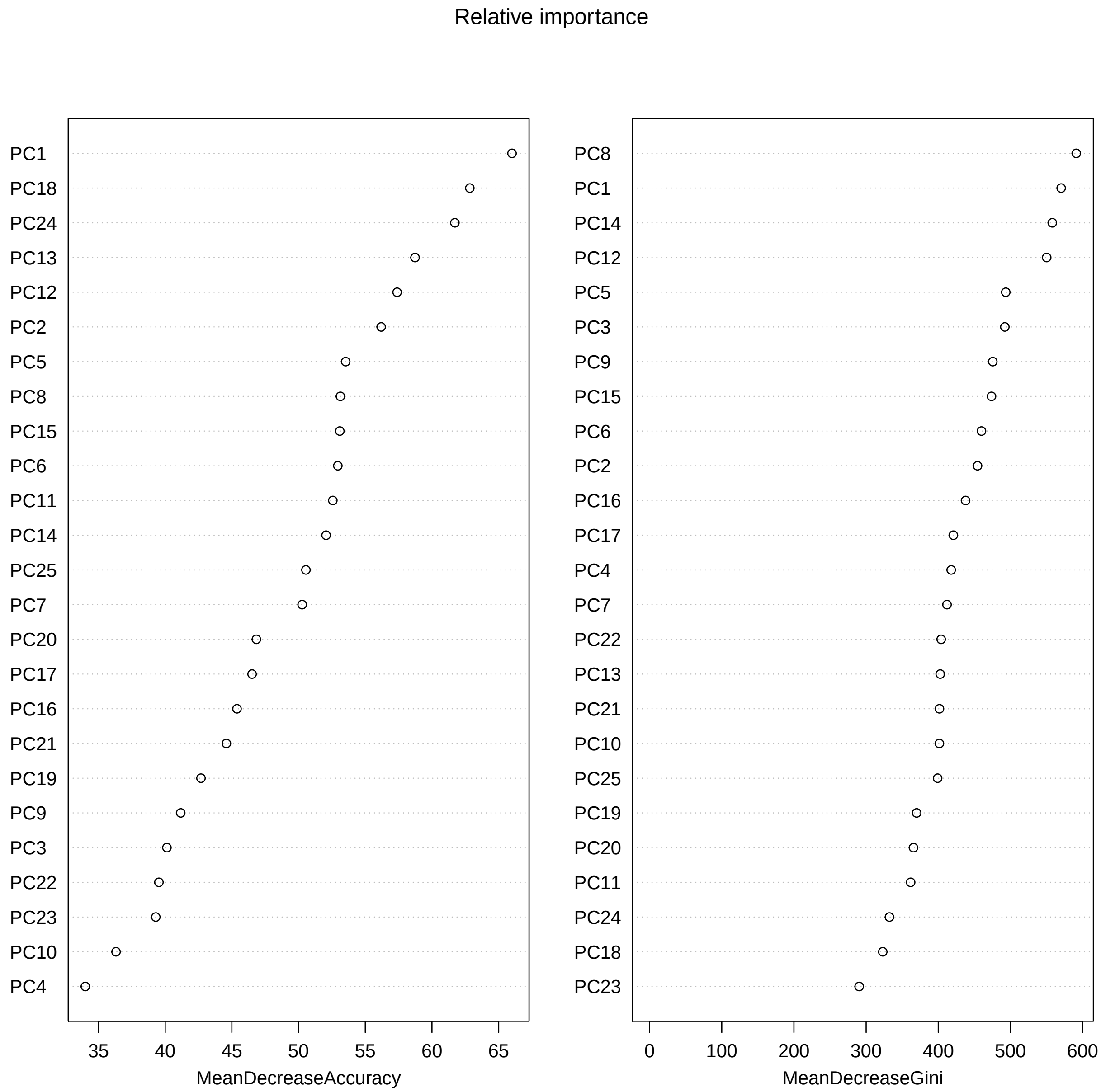
```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

Relative importance of principal components

```
varImpPlot(modelFit$finalModel, sort = TRUE, main = "Relative importance")
```



Cross Validation and Out of Sammple Error

Result below shows that model is 97% accurate against the validation data.

```
confusionM <- confusionMatrix(validationData$classe, predict(modelFit, validationPC))
confusionM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1663    6    4    1    0
##          B   22 1097   17    0    3
```

```
##          C    2   18  990   13    3
##          D    1    1   48  911    3
##          E    0    5    5    9 1063
##
## Overall Statistics
##
##                Accuracy : 0.973
##                  95% CI : (0.968, 0.977)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.965
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.985    0.973    0.930    0.975    0.992
## Specificity             0.997    0.991    0.993    0.989    0.996
## Pos Pred Value          0.993    0.963    0.965    0.945    0.982
## Neg Pred Value          0.994    0.994    0.985    0.995    0.998
## Prevalence              0.287    0.192    0.181    0.159    0.182
## Detection Rate          0.283    0.186    0.168    0.155    0.181
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.991    0.982    0.961    0.982    0.994
```

out of sample error is about 0.027

```
error <- 1 - as.numeric(confusionM$overall[1]);
error
```

```
## [1] 0.02736
```

Test

We apply our model to test data and get the results below

```
result <-predict(modelFit,testPC)
result
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```