Executive Summary

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of six participants. The objective is to use this data from the six participants in this supervised learning task of prediction to determine the efficiency of their exercise.

Load the Libraries

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.1.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3

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

## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.1.3
```

```
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.1.3

##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Dataset

Two files were provided, for training and test.

```
# read the csv file for training
trainSet <- read.csv("./data/pml-training.csv", na.strings= c("NA","","  "))
```

```
# preprocessing (cleaning data)
cleanData1 <- apply(trainSet, 2, function(x) {sum(is.na(x))})
cleanData2 <- trainSet[,which(cleanData1 == 0)]

# discarded first 8 columns which were irrelevant to building the model.
cleanData <-cleanData2[8:length(cleanData2)]
```

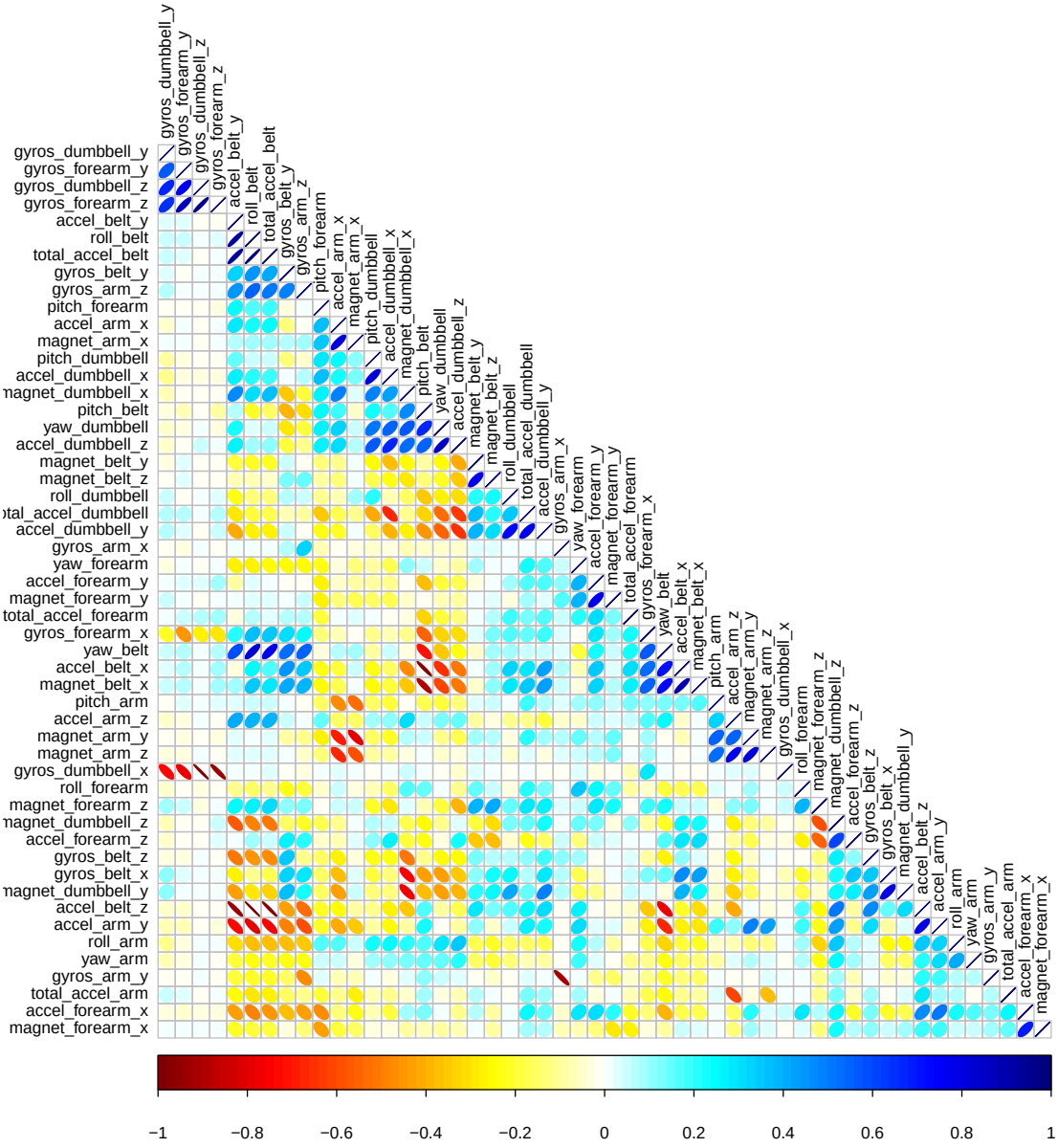
Model Generation

The training datas split into training and validation sets using a 75, 25 respective split. Training is for training the model and validation is to validate the training model.

```
# breakup training data into validation and training data
train <- createDataPartition(y = cleanData$classe, p = 0.75, list = FALSE)
training <- cleanData[train, ]
validation <- cleanData[-train, ]
```

First a corelation plot is created for the 52 variables and then plotted.

```
#generate and plot correlation matrix
COR<-cor(training[, -length(training)], method = "pearson")
col1 <- colorRampPalette(c("#7F0000","red","#FF7F00","yellow","white",
        "cyan", "#007FFF", "blue","#00007F"))
corrplot(COR, order="hclust", addrect=3,method = "ellipse", col=col1(100),type = "lower", tl.cex = 0.8,
```



The red clusters in the plot represent negative correlation and blue clusters represe4nt postive correlation.

For classification random forest classifier was utilized.

```
# random forest model is fitted using all of the variables in the training dataset.
rf_model <- randomForest(classe ~ ., data = training, ntree=1000)
rf_model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 7
##
```

```
##         OOB estimate of  error rate: 0.47%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4178    6    1    0    0 0.001672640
## B   15 2830    3    0    0 0.006320225
## C    0   14 2551    2    0 0.006232957
## D    0    0   22 2388    2 0.009950249
## E    0    0    0    4 2702 0.001478197
```

The error on training is fairly low, the diagonal shows the correct classification. Now the model is tested against the validation set:

```
crossValidation <- predict(rf_model, validation)
confusionMatrix(validation$classe, crossValidation)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    8  938    3    0    0
##          C    0    9  844    2    0
##          D    0    0    8  795    1
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.9903, 0.9952)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9905   0.9871   0.9938   0.9989
## Specificity            1.0000   0.9972   0.9973   0.9978   0.9993
## Pos Pred Value         1.0000   0.9884   0.9871   0.9888   0.9967
## Neg Pred Value         0.9977   0.9977   0.9973   0.9988   0.9998
## Prevalence             0.2861   0.1931   0.1743   0.1631   0.1833
## Detection Rate         0.2845   0.1913   0.1721   0.1621   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9971   0.9939   0.9922   0.9958   0.9991
```

At 99.82%, the model performed very well on validation set. Now will test on test set.

cleanData1 <- apply(trainSet, 2, function(x) {sum(is.na(x))}) cleanData2 <- trainSet[,which(cleanData1 == 0)] # discarded first 8 columns which were irrelevant to building the model. cleanData <- cleanData2[8:length(cleanData2)]

```r
# test data (Note: same thing done to training for cleaning also done to test)
testSet<- read.csv("./data/pml-testing.csv", na.strings= c("NA","","  "))
testClean1<- apply(testSet, 2, function(x) {sum(is.na(x))})
testClean2 <- testSet[,which(testClean1== 0)]
testClean <- testClean2[8:length(testClean2)]

#test model on test set
testModel <- predict(rf_model, testClean )
testModel
```

```
##  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
```

Conclusion

Remarkably the model is very good on this data and it's reasonable to assume that the out of sample error will be very good as well.