March 26, 2016

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

Practical Machine Learning Assignment

Summary

We intend to use the data from the fitness tracker Jawbone to predict the activities people were doing while wearing the device. We find that we can find a very accurate prediction model by using random forest method.

Exploratory Data Analysis

First we will load the training and the testing data. These can be found online here and here.

The training data has 19622 observations and the testing only has 20.

```
rawtraining<-read.csv("training.csv",na.strings=c("", "NA", "NULL"))
rawtesting<-read.csv("testing.csv",na.strings=c("", "NA", "NULL"))
dim(rawtraining)
```

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

```
dim(rawtesting)
```

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

Now with 160 variables many of these will not be useful for our prediction. The first step is to remove and variable that contains NAs. The second is to remove any data that does not help describe motion. These would be things like user name and time stamps.

```
nacheck <- apply(rawtraining,2,function(x){ sum(is.na(x)) })
keeptraining <- subset(rawtraining[,which(nacheck ==0)], select=-c(X, user_name, new_window, num_window
dim(keeptraining)
```

```
## [1] 19622    53
```

Since the rawtesting data set provided is quite small and the rawtraining set very large we should consider the given rawtesting set the final testing set and partition the keeptraining set into a training and testing set for the purposes of building the model

```
set.seed(202020)
inTrain <- createDataPartition(keeptraining$classe, p=0.7, list=F)
training <- keeptraining[inTrain,]
testing <- keeptraining[-inTrain,]
```

Model building

Now we have explored the data and removed variables we don't need. The data set is very large so we want to be careful setting up our model. First we should set some controls on the random forest model so it is quicker (this does sacrifice some accuracy but it will still be). I choose to use the cross validation method and limited it to 4 folds. I also let r take advantage of my multi-core computer.

```
ctrl <- trainControl(allowParallel=T, method="cv", number=4)
rfmodel <- train(classe~., data = training, method="rf", trControl=ctrl)
```

```
## Loading required package: randomForest

## 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
```

```
rfpred <- predict(rfmodel,newdata = testing)
confusionMatrix(testing$classe, rfpred)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    9 1125    4    1    0
##          C    0    2 1023    1    0
##          D    0    0   13  949    2
##          E    0    1    1    0 1080
```

```
confusionMatrix(rfpred,testing$classe)$overall[1]
```

```
##  Accuracy
## 0.9938828
```

As you can see from the matrix the prediction is very accurate and missed very few of the class of exercises. When we look at accuracy we see it is ~99.39% accurate.

Now let us test this model verses the original test model, first we will have to remove the same columns we removed from the rawtraining data.

```
keeptesting <- subset(rawtesting[,which(nacheck ==0)], select=-c(X, user_name, new_window, num_window, 
rfpredrawtest <- predict(rfmodel,newdata = keeptesting)
```

Now these are the answers to the quiz questions. When submitted it came back with a perfect score.

Conclusion

This model was very accurate at prediction the answers to the test sets questions but I would estimate that in a data set we would not see the same 99% accuracy and it would be lower.