Work Out Data Machine Learning Project

The goal of our machine learning project to create a model that can accurately predict what class of working out an individual is in. To do this we are suppose to use machine learning functions and principals to make the optimal model for predicting. ## 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).

Loading and Cleaning the Data

First, lets download the data from the links below

```
train_url <- download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "tra
test_url <- download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "testing.csv", "testing.c
```

Now, let's load the data and write the 'na.strings' argument in so that we change confounding null and na values in as NA so that we may get rid of them.

```
training <- read.csv("training_dat.csv", header = TRUE, na.strings=c("NA", "#DIV/0!",""))
testing <- read.csv("testing_dat.csv", header = TRUE, na.strings=c("NA", "#DIV/0!",""))</pre>
```

Next, let's change the column sum of the na values into 0 so that we can eliminate NA values.

```
training <- training[,colSums(is.na(training)) == 0]
testing <- testing[,colSums(is.na(training)) == 0]</pre>
```

After that, let's remove the first seven columns since they hold data types that will complicate the model.

```
training <- training[,-c(1:7)]
testing <- testing [,-c(1:7)]</pre>
```

Data Partioning and Model Building

Next, we will download and implement the 'caret' package to split the cleaned data into training and testing sets. These are the sets we will use to run the model.

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

inTrain <- createDataPartition(training$classe, p = 0.6, list = FALSE)

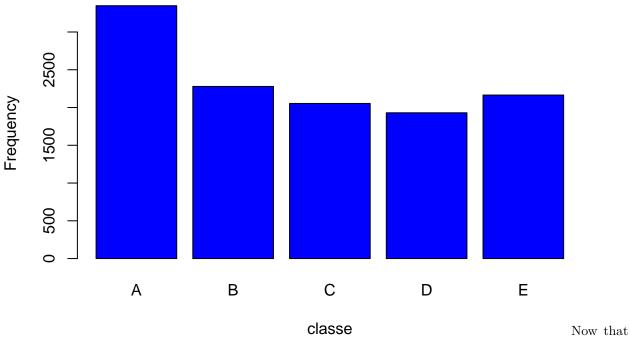
train <- training[inTrain,]

test <- training[-inTrain,]</pre>
```

With a histogram we can create a histogram of the frequency of each of the "classe"s we will be testing for

```
plot(train$classe, col = "blue", main="bar chart of 'classe' levels in train", xlab = "classe", ylab = "...")
```

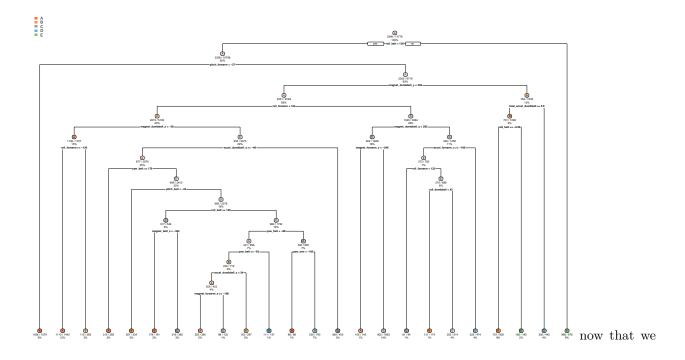
bar chart of 'classe' levels in train



we have our data split we can create our models ### create models We are going to use the rpart model to create our model. Using the 'rpart.plot' function in the 'rpart.plot' package, we can create a decision tree that will tell us how the model was created.

```
library(rpart); library(rpart.plot)
model1 <- rpart(classe~., data = train, method = "class")
rpart.plot(model1, main = "Classification", extra = 102, under = TRUE, faclen=0)</pre>
```

Classification



have our model lets make predictions using the model and the test data and see how accurate the model is at predicting using confusion matrix

```
pred1 <- predict(model1, test, type = "class")</pre>
confusionMatrix(pred1, test$classe)
## Confusion Matrix and Statistics
##
##
             Reference
                 Α
                            C
                                  D
                                       Ε
## Prediction
                       В
##
            A 2012
                     246
                           32
                                 58
                                      30
##
            В
                 88
                     875
                           77
                                121
                                     165
            C
                 64
                     228 1149
                                204
                                     186
##
            D
##
                 46
                     111
                           77
                                828
                                     124
##
            Ε
                 22
                      58
                           33
                                 75
                                     937
##
  Overall Statistics
##
                   Accuracy : 0.7394
##
##
                     95% CI: (0.7295, 0.749)
##
       No Information Rate: 0.2845
       P-Value [Acc > NIR] : < 2.2e-16
##
##
##
                      Kappa: 0.6698
##
    Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##
                         Class: A Class: B Class: C Class: D Class: E
## Sensitivity
                           0.9014
                                     0.5764
                                               0.8399
                                                        0.6439
                                                                  0.6498
## Specificity
                                     0.9287
                                               0.8947
                                                        0.9454
                                                                  0.9706
                           0.9348
## Pos Pred Value
                           0.8461
                                     0.6599
                                               0.6275
                                                        0.6981
                                                                  0.8329
## Neg Pred Value
                           0.9598
                                     0.9014
                                               0.9636
                                                        0.9312
                                                                  0.9249
## Prevalence
                           0.2845
                                     0.1935
                                               0.1744
                                                        0.1639
                                                                  0.1838
## Detection Rate
                           0.2564
                                     0.1115
                                               0.1464
                                                        0.1055
                                                                  0.1194
## Detection Prevalence
                           0.3031
                                               0.2334
                                                                  0.1434
                                     0.1690
                                                        0.1512
                                     0.7526
                                                        0.7946
## Balanced Accuracy
                           0.9181
                                               0.8673
                                                                  0.8102
```

as we can see from the confusion matrix there is very low accuracy in the model. Lucky for us we can us the 'randomForest' to create a bunch of models for us and choose the best one based on the outputs. ### prediction w/ random forest

```
library("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

model2 <- randomForest(classe~., data = train, method = "class")</pre>
```

Now that we have our model let's predict to see how accurate it is.

```
confusionMatrix(pred2, test$classe)
## Confusion Matrix and Statistics
##
##
             Reference
## Prediction
                 Α
                       В
                            C
                                 D
                                      Ε
                       6
                                       0
##
            A 2231
                            0
                                 0
##
            В
                 0 1503
                           10
                                       0
                                 0
            С
##
                 0
                       9 1357
                                17
                                       1
                            1 1269
##
            D
                 0
                       0
                                       3
##
            Ε
                       0
                            0
                                 0 1438
##
## Overall Statistics
##
##
                  Accuracy : 0.9939
                     95% CI : (0.9919, 0.9955)
##
##
       No Information Rate: 0.2845
       P-Value [Acc > NIR] : < 2.2e-16
##
##
##
                      Kappa: 0.9923
##
   Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##
                         Class: A Class: B Class: C Class: D Class: E
## Sensitivity
                           0.9996
                                    0.9901
                                              0.9920
                                                       0.9868
                                                                 0.9972
## Specificity
                                    0.9984
                                              0.9958
                                                       0.9994
                                                                 0.9998
                           0.9989
## Pos Pred Value
                           0.9973
                                    0.9934
                                              0.9805
                                                       0.9969
                                                                 0.9993
## Neg Pred Value
                           0.9998
                                    0.9976
                                              0.9983
                                                       0.9974
                                                                 0.9994
## Prevalence
                           0.2845
                                    0.1935
                                              0.1744
                                                       0.1639
                                                                 0.1838
## Detection Rate
                           0.2843
                                    0.1916
                                              0.1730
                                                       0.1617
                                                                 0.1833
## Detection Prevalence
                           0.2851
                                    0.1928
                                              0.1764
                                                       0.1622
                                                                 0.1834
                                                                 0.9985
## Balanced Accuracy
                           0.9992
                                    0.9943
                                              0.9939
                                                       0.9931
```

pred2 <- predict(model2, test)</pre>

As we can see this model is extremly and accurate thanks to the random forest model. This model will be the model that we use in the future to predict the workout class of new observations.