Final Project Report - Practical Machine Learning Course

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.

Project Goals

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Peer Review Porter

1.Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Reproduceablity

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

```
library(corrplot)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

Loading required package: lattice

Loading required package: ggplot2

Loading the same seed

```
set.seed(23456)
```

Getting the data

Load data to memory for use in project

```
training<- read.csv("pml-training.csv", header = TRUE, na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", header = TRUE, na.strings = c("NA", ""))
```

Columns in the orignal training and testing datasets that are mostly filled with missing values are then removed.

Our updated training dataset now has fewer variables to review in our analysis.

```
rcol <- colSums(is.na(training))
rcol_log <- (rcol == 0)
training_rcol <- training[, (colSums(is.na(training)) == 0)]
testing <- testing[, (colSums(is.na(training)) == 0)]
```

Create another logical vector in order to delete additional unnecessary columns from the pared-down training and testing datasets.

```
dcols_log <- grepl("X|user_name|timestamp|new_window", colnames(training_rcol))
training_rcol <- training_rcol[, !dcols_log]
testing_rcol <- testing[, !dcols_log]
```

Partioning the training set two

Partioning training data set into two data set, 70% for iTraining, 30% for iTesting

```
inTrain <- createDataPartition(y=training_rcol$classe, p=0.7, list = FALSE)
iTraining <- training_rcol[inTrain,]
iTesting <- training_rcol[-inTrain,]
dim(iTraining)
```

```
## [1] 13737    54
```

```
dim(iTesting)
```

```
## [1] 5885   54
```

Cleaning the data

The following transformations were used to clean the data:

Cleaning NearZeroVariance Variables Run this code to view possible NZV Variables:

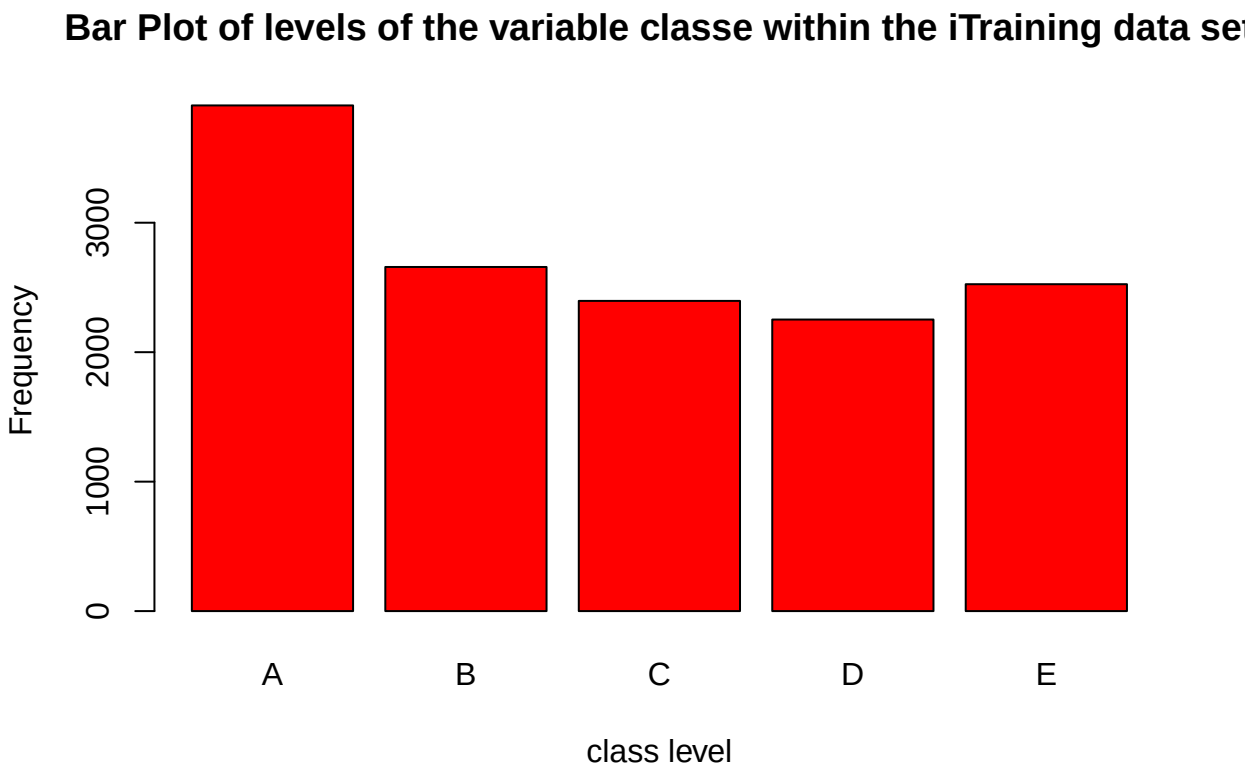
```
dataNZV <- nearZeroVar(iTraining, saveMetrics = TRUE)
dim(dataNZV)
```

```
## [1] 54  4
```

Plot of Data variables

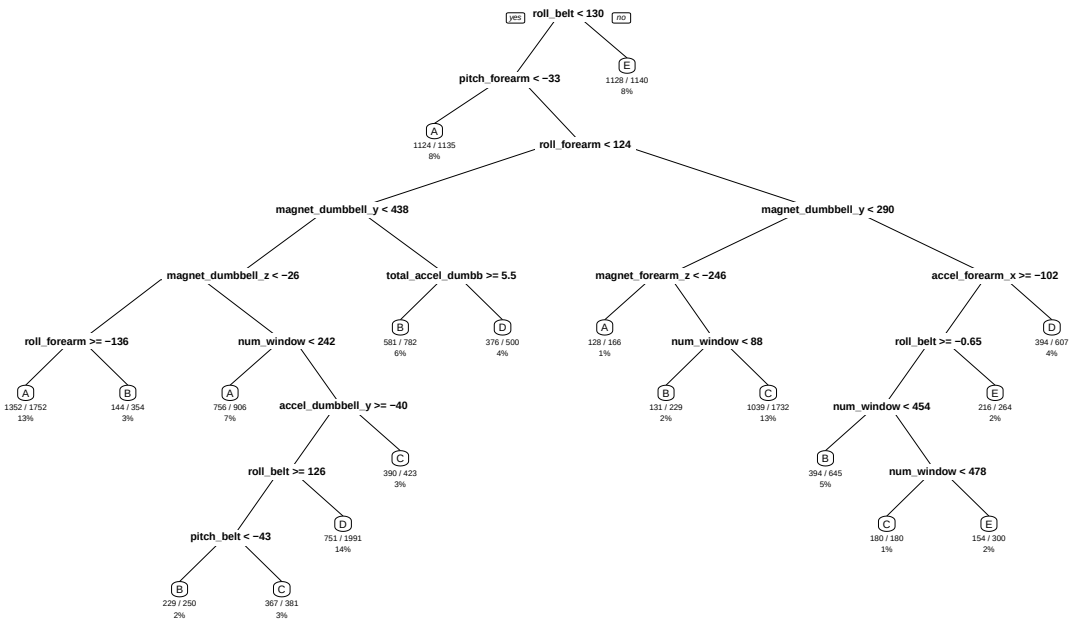
The variable “classe” contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the subTraining data set and compare one another.

```
plot(iTraining$classe, col="red",
     main = "Bar Plot of levels of the variable classe within the iTraining data set",
     xlab="class level", ylab="Frequency")
```



First prediction model: Using Decision Tree

```
modFit <- rpart(classe ~.,data=iTraining, method="class")

predictions <- predict(modFit, iTesting, type = "class")

rpart.plot(modFit,main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Cross Validation

Call the 'predict' function again so that our trained model can be applied to our cross validation test dataset.

```
confusionMatrix(predictions, iTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1425  145   11   48   60
##          B   73  619   19   87  168
##          C   24   68  864  138   76
##          D  147  258  130  652  133
##          E    5   49    2   39  645
##
## Overall Statistics
##
##                Accuracy : 0.7145
##                  95% CI : (0.7028, 0.726)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6396
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8513   0.5435   0.8421   0.6763   0.5961
## Specificity            0.9373   0.9269   0.9370   0.8643   0.9802
## Pos Pred Value         0.8437   0.6408   0.7385   0.4939   0.8716
## Neg Pred Value         0.9407   0.8943   0.9656   0.9317   0.9151
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2421   0.1052   0.1468   0.1108   0.1096
## Detection Prevalence   0.2870   0.1641   0.1988   0.2243   0.1257
## Balanced Accuracy      0.8943   0.7352   0.8896   0.7703   0.7882
```

Second prediction model: Using Random Forest

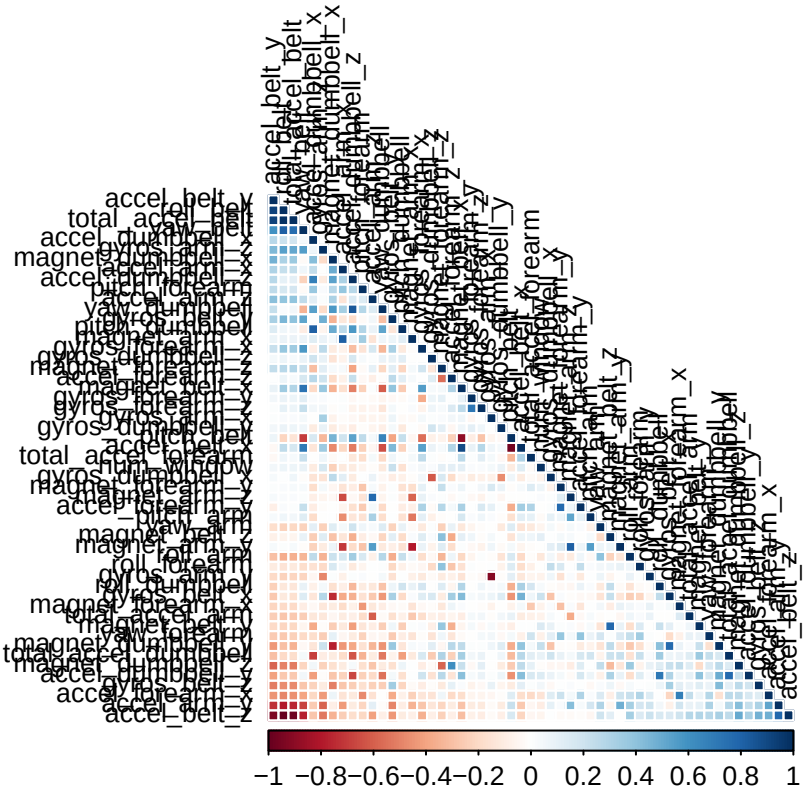
```
modFit1 <- randomForest(classe ~., data=iTraining)

predictions1 <- predict(modFit1, iTesting, type="class")

confusionMatrix(predictions1, iTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    0 1138    1    0    0
##          C    0    0 1025    3    0
##          D    0    0    0  961    1
##          E    1    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9991   0.9990   0.9969   0.9991
## Specificity            0.9998   0.9998   0.9994   0.9998   0.9998
## Pos Pred Value         0.9994   0.9991   0.9971   0.9990   0.9991
## Neg Pred Value         0.9998   0.9998   0.9998   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1934   0.1742   0.1633   0.1837
```

```
## Detection Prevalence   0.2845   0.1935   0.1747   0.1635   0.1839
## Balanced Accuracy      0.9996   0.9995   0.9992   0.9983   0.9994
```

The modified data set contain 54 variables, with the last column containing "classe" variable trying to predict.

```
trainingMat <- cor(iTraining[, -54])
corrplot(trainingMat, order = "FPC", method = "color", type = "lower", tl.cex = 0.8,
         tl.col = rgb(0, 0, 0))
```



Results for Submission

```
#Predict the outcome results for submission
predictResult <- predict(modFit1, newdata = testing, type="class")
predictResult
```

```
##  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
```

```
#Write results to file for submission
pml_write_files = function(x){
  n = length(x)
```

```
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictResult)
```