Machine Learning Final Project

Summary

This report builds a predictive model for personal activity and how much of a particular activity an individual does; in particular, how well they do it (quality). Quality is broken down into five categories and data is from accelerometers on participants on the belt, forearm, arm, and dumbbell of six participants.

Step 1: Upload training dataset and Set seed.

Here we are loading required (and potentially required) packages.

```
install.packages(
  "ggplot2",
  repos = c("http://rstudio.org/_packages",
          "http://cran.rstudio.com")
library(caret)
library(AppliedPredictiveModeling)
library(randomForest)
library(RColorBrewer)
library(rattle)
library(rpart)
library(kernlab)
library(rpart.plot)

set.seed(12345)

# Upload training dataset
  training = read.csv("training data for machine learning project",
                header=TRUE,
                stringsAsFactors=FALSE,
                na.strings=c("NA", "N/A"))
  dim(training)
dim(training)
[1] 19622   160
```

Step 2: Cleaning and Partitioning training data.

Here we are cleaning and partitioning the training dataset which we imported in step 1. There were a lot of NA entries in the raw training set, with just over 400 cases being complete. So what I did is remove the NA cases and replace with 0. I also converted all columns to numeric. Further down in step 2, what I did was randomly divide the training dataset into two sets, with 75% as the training subset and 25% as the test subset.

##clean predictor variables – training

```
> #clean predictor variables for training data
> cldata <- training[,-c(1:7, 160)] # drop first 7 and last columns
> cldata<- suppressWarnings(data.frame(apply(cldata, 2, as.numeric))) # conve
rt all columns to numeric
> cldata[is.na(cldata)] <- 0; sum(is.na(cldata)) # replace <NA> with 0
[1] 0
> cldata <- cldata[vapply(cldata, function(x) length(unique(x)) > 1, logical(
1L))] # remove same value columns
```

```
dim(cldata)

[1] 19622   143
```

#select training with clean data

training <- data.frame(classe=training$classe, cldata); rm(cldata)

training$classe <- as.factor(training$classe)

#Randomly divide training data in to two sets: 75% as “train” subset, and 25% as “test” subset.

```
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
> myTraining <- training[inTrain, ];
> myTesting <- training[-inTrain, ]
> dim(myTraining); dim(myTesting)
[1] 14718   144
[1] 4904  144
```

Step 3: Run models and test accuracy with the 4 “test” subset data

I decided to run Random Forest with PCA for feature selection, as well as Random Forest without PCA. Then I ran Classification with and without PCA, and finally Boosted Tree with PCA. By running these models I can then test the subset data and compare accuracy to find the best model fit for the data.

# PCA for Feature Selection

```r
prePCA <- preProcess(myTraining, method=c("center", "scale", "pca"), thresh = 0.8); prePCA
trainPCA <- predict(prePCA, myTraining)
summary(trainPCA)

# Random Forest with PCA
accRFPCA <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
rfpca <- randomForest(classe~., data=trainPCA, method="class")
test = myTesting
testrfpca <- predict(prePCA, test)
predrfpca <- predict(rfpca, testrfpca)
accRFPCA[i,1] <- confusionMatrix(test$classe, predrfpca)$overall["Accuracy"]
}; rm(rfpca, test, testrfpca, predrfpca)
##variable importance for RF+PCA
importance(rfpca)
varImp(rfpca)

# Random Forest without PCA
accRF <- matrix(0, nrow=4, ncol=1)
RF <- randomForest(classe~., data=myTraining, method="class")
for (i in 1:4) {
  test = myTesting
  predRF <- predict(RF, test)
  accRF[i,1] <- confusionMatrix(test$classe, predRF)$overall["Accuracy"]
}; rm(RF, test, predRF)
##variable importance for RF
importance(RF)
varImp(RF)
```

```
#Classication rpart+ PCA
cl_rpt_pca <- train(classe~., data=trainPCA, method="rpart")
fancyRpartPlot(cl_rpt_pca$finalModel, sub="")
acccl_rpt_pca <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test=myTesting
testcl_rpt_pca <- predict(prePCA, test)
predcl_rpt_pca <- predict(cl_rpt_pca, testcl_rpt_pca)
acccl_rpt_pca[i,1] <- confusionMatrix(test$classe, predcl_rpt_pca)$overall["Accuracy"]
}; rm(cl_rpt_pca, test, testcl_rpt_pca, predcl_rpt_pca)

#Classification
dfrpart <- train(classe~., data=myTraining, method="rpart")
fancyRpartPlot(dfrpart$finalModel, sub="")
accdfrpt <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
  test=myTesting
  preddfrpart <- predict(dfrpart, test)
  accdfrpt[i,1] <- confusionMatrix(test$classe, preddfrpart)$overall["Accuracy"]
}; rm(dfrpart, test, preddfrpart)

##Classification (rpart= "class")
clrpart <- rpart(classe~., data=myTraining, method="class")
fancyRpartPlot(clrpart, sub="")
accclrpart <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
  test=myTesting
  predclrpart <- predict(clrpart, test, type ="class")
  accclrpart[i,1] <- confusionMatrix(test$classe, predclrpart)$overall["Accuracy"]
}; rm(clrpart, test, predclrpart)
```

#Boosted Tree with PCA

```
BPCA <- train(classe~., method="gbm", data=trainPCA)
quiet(expr, all = TRUE)
accGBM <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
  test = myTesting
  testGBM <- predict(prePCA, test)
  predGBM <- predict(BPCA, testGBM)
  accGBM[i,1] <- confusionMatrix(test$classe, predGBM)$overall["Accuracy"]
}; rm(BPCA, test, testGBM, predGBM, i)
quiet(expr, all = TRUE)
```

Step 5: Compare speed and accuracy of each potential model, then choose one to apply to test data.

What we see below is the validation and comparison for each of the tested models. It is apparent that Random Forests and Random Forests with PCA are the best models to apply to the test data in terms of speed and accuracy. I have decided to use Random Forests given the result of accuracy and speed being the best as compared to the other methods tested. The cross validation used to compare the different tests make me confident in this decision and that RF will have the lowest chance of error (less than 1%).

```
> acc_all <- data.frame(accRFPCA, accRF, acccl_rpt_pca, accdfrpt, accclrpart,
accGBM)
> names(acc_all) <- c("RF+PCA", "RF", "rpart+PCA", "rpart(default)", "rpart(c
lass)", "gbm+PCA"); acc_all
     RF+PCA        RF rpart+PCA rpart(default) rpart(class)   gbm+PCA
1 0.9600326 0.9946982 0.3882545       0.487969    0.7310359 0.7585644
2 0.9612561 0.9946982 0.3882545       0.487969    0.7310359 0.7585644
3 0.9616639 0.9946982 0.3882545       0.487969    0.7310359 0.7585644
4 0.9614600 0.9946982 0.3882545       0.487969    0.7310359 0.7585644
> colMeans(acc_all)
        RF+PCA             RF      rpart+PCA rpart(default)   rpart(class)
gbm+PCA
     0.9611032      0.9946982      0.3882545      0.4879690      0.7310359
0.7585644
```

Step 6: Predict on test data using selected model

Here the test set is uploaded and cleaned. Just as with the training set, the test set columns were converted to numeric values, and the NA columns were removed and replaced. As we see below doing the prediction with the test data, using Random Forest, I have predicted the manner in which the exercises were done for 20 different test cases. What the prediction shows is that for 7 cases level A is predicted; for 8 cases, level B; for 1 case, level C; 1 case, level D; and 3 cases, level E.

```
> #Upload and clean test data
> testing = read.csv("testing data for machine learning project.csv",
+                    header = TRUE,
+                    stringsAsFactors=FALSE,
+                    na.strings=c("NA", "N/A"))
> cldata2 <- testing[,-c(1:7, 160)] ## drop the first 7 and the last columns
> cldata2<- suppressWarnings(data.frame(apply(cldata2, 2, as.numeric))) ## co
nvert all columns to numeric
> cldata2[is.na(cldata2)] <- 0; sum(is.na(cldata2)) ## replace <NA> with 0
[1] 0
> dim(testing)
[1]  20 160
```

# Predict using test data

```
> testing <- data.frame(classe="", cldata2); rm(cldata2)
> testRF <- randomForest(classe~. , data=myTraining, method="class")
> pred <- predict(testRF, testing); pred
 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
```

FIGURES

Figure 1: Classification – Default rpart()

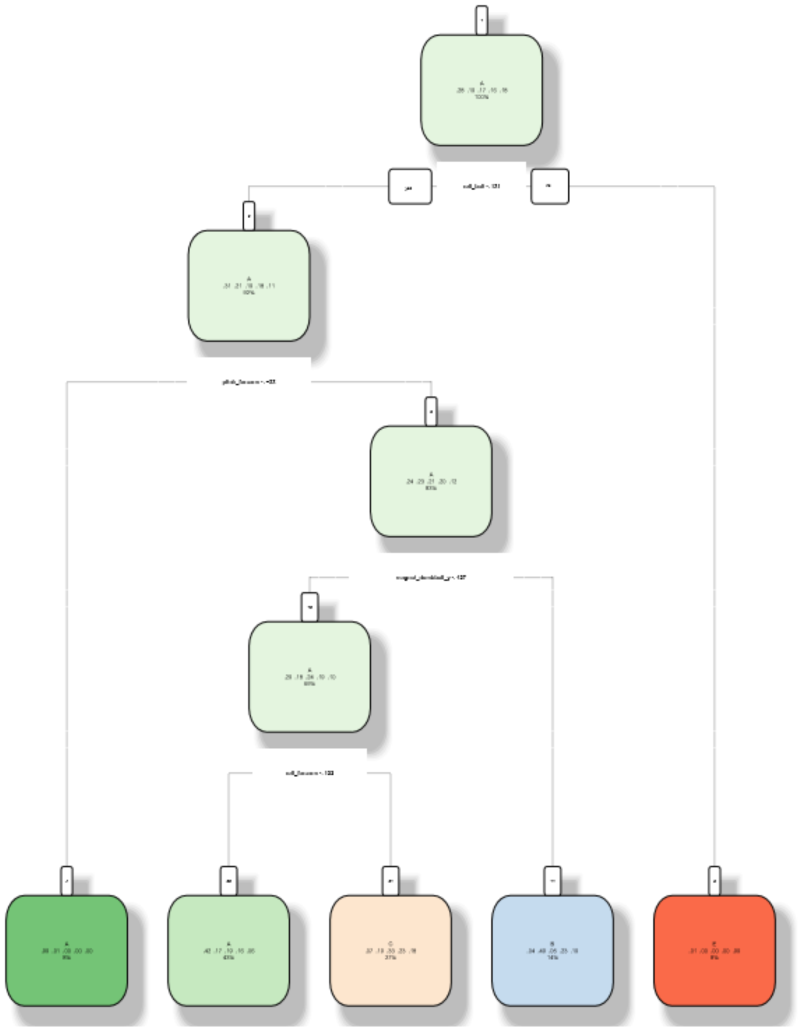
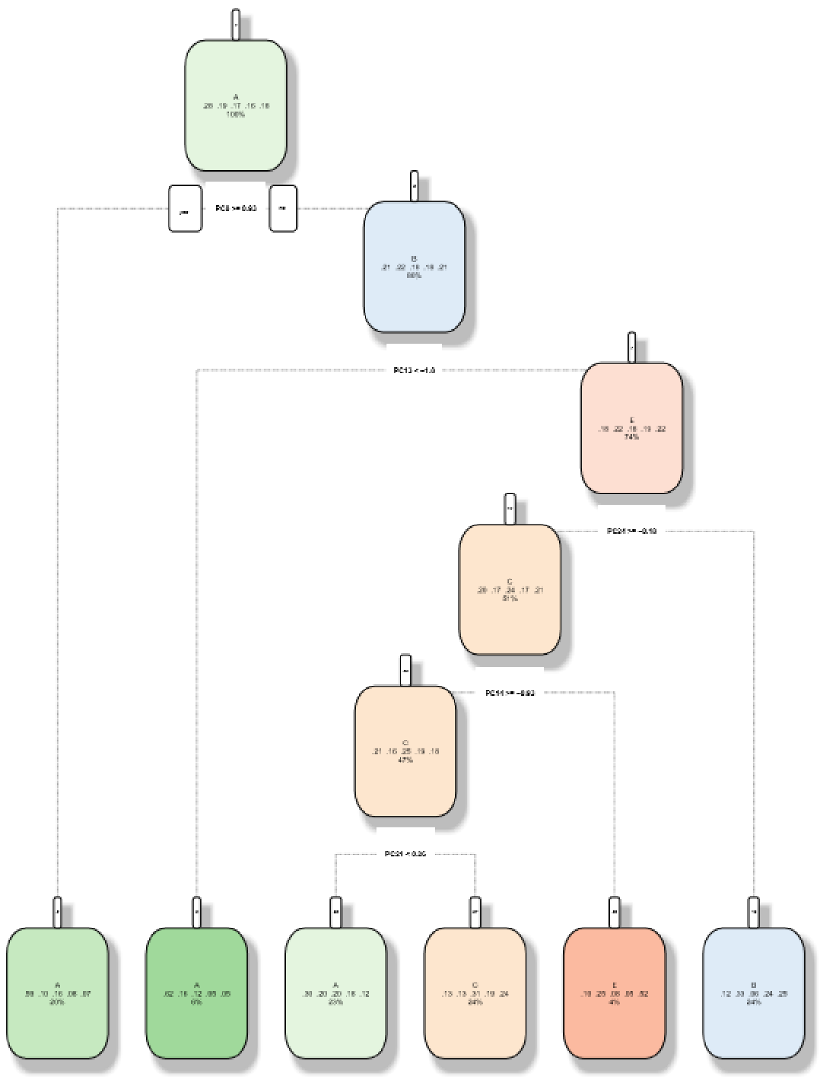


Figure 2: Random Forrest + PCA



A
.26 .19 .17 .16 .18
100%
yes
PC8 >= 0.93
no
B
.21 .22 .16 .18 .21
80%
PC13 < -1.0
E
.18 .22 .18 .19 .22
74%
PC24 >= -0.10
C
.20 .17 .24 .17 .21
51%
PC11 >= -0.93
C
.21 .16 .25 .19 .18
47%
PC21 < 0.36
A
.58 .10 .16 .08 .07
20%
A
.62 .16 .12 .05 .05
6%
A
.30 .20 .20 .18 .12
23%
C
.13 .13 .31 .19 .24
24%
E
.10 .25 .08 .05 .52
4%
B
.12 .33 .06 .24 .25
24%