Predicting manner in which excersize was done

Course: Practical Machine Learning

By: Mindaugas Mozuraitis

Data Processing

Step 1: Loading required packages

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.4
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.4.4
```

```
## Loading required package: bitops
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.4.4
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
library(caretEnsemble)
```

```
## Warning: package 'caretEnsemble' was built under R version 3.4.4
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

Step 2: Downloading and reading in the data

```
URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
x <- getURL(URL)
train <- read.csv(textConnection(x), na.strings=c("", " ", "NA", "#DIV/0!"))

testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testx <- getURL(testURL)
test <- read.csv(textConnection(testx),na.strings=c("", " ", "NA", "#DIV/0!"))
```

Step 3: Partitioning data into training and validation sets

```
inTrain=createDataPartition(y=train$classe, p=0.7, list=FALSE)
train0=train[inTrain,]
test0=train[-inTrain,]
dim(train0)
```

```
## [1] 13737   160
```

Step 4: Removing variables where more than 50% of values are missing

```
train_summary=as.data.frame(summary(train0))
train_summary1=train_summary[grep("NA's", train_summary$Freq),]
train_summary1$Freq=gsub("NA's", "", train_summary1$Freq)
train_summary1$Freq=gsub(":", "", train_summary1$Freq)
train_summary1$Freq=gsub(" ", "", train_summary1$Freq)
train_summary1$PCNT_NA=as.numeric(train_summary1$Freq)/nrow(train0)
train_summary1
```

```
##      Var1                    Var2  Freq   PCNT_NA
## 84             kurtosis_roll_belt 13445 0.9787435
## 91            kurtosis_picth_belt 13463 0.9800539
## 93              kurtosis_yaw_belt 13737 1.0000000
## 105            skewness_roll_belt 13445 0.9787435
## 112          skewness_roll_belt.1 13463 0.9800539
## 114             skewness_yaw_belt 13737 1.0000000
## 126                 max_roll_belt 13439 0.9783068
## 133                max_picth_belt 13439 0.9783068
## 140                  max_yaw_belt 13445 0.9787435
## 147                 min_roll_belt 13439 0.9783068
## 154                min_pitch_belt 13439 0.9783068
## 161                  min_yaw_belt 13445 0.9787435
## 168           amplitude_roll_belt 13439 0.9783068
## 175          amplitude_pitch_belt 13439 0.9783068
## 182            amplitude_yaw_belt 13445 0.9787435
## 189          var_total_accel_belt 13439 0.9783068
## 196                 avg_roll_belt 13439 0.9783068
## 203              stddev_roll_belt 13439 0.9783068
## 210                 var_roll_belt 13439 0.9783068
## 217                avg_pitch_belt 13439 0.9783068
## 224             stddev_pitch_belt 13439 0.9783068
## 231                var_pitch_belt 13439 0.9783068
## 238                  avg_yaw_belt 13439 0.9783068
## 245               stddev_yaw_belt 13439 0.9783068
## 252                  var_yaw_belt 13439 0.9783068
## 350                 var_accel_arm 13439 0.9783068
## 357                  avg_roll_arm 13439 0.9783068
## 364               stddev_roll_arm 13439 0.9783068
## 371                  var_roll_arm 13439 0.9783068
## 378                 avg_pitch_arm 13439 0.9783068
## 385              stddev_pitch_arm 13439 0.9783068
## 392                 var_pitch_arm 13439 0.9783068
## 399                   avg_yaw_arm 13439 0.9783068
## 406                stddev_yaw_arm 13439 0.9783068
## 413                   var_yaw_arm 13439 0.9783068
## 483             kurtosis_roll_arm 13492 0.9821650
## 490            kurtosis_picth_arm 13494 0.9823105
## 497              kurtosis_yaw_arm 13448 0.9789619
## 504             skewness_roll_arm 13492 0.9821650
## 511            skewness_pitch_arm 13494 0.9823105
## 518              skewness_yaw_arm 13448 0.9789619
## 525                  max_roll_arm 13439 0.9783068
## 532                 max_picth_arm 13439 0.9783068
## 539                   max_yaw_arm 13439 0.9783068
## 546                  min_roll_arm 13439 0.9783068
## 553                 min_pitch_arm 13439 0.9783068
## 560                   min_yaw_arm 13439 0.9783068
## 567            amplitude_roll_arm 13439 0.9783068
## 574           amplitude_pitch_arm 13439 0.9783068
## 581             amplitude_yaw_arm 13439 0.9783068
## 609        kurtosis_roll_dumbbell 13441 0.9784524
## 616       kurtosis_picth_dumbbell 13440 0.9783796
```

```
## 618          kurtosis_yaw_dumbbell 13737 1.0000000
## 630         skewness_roll_dumbbell 13441 0.9784524
## 637        skewness_pitch_dumbbell 13440 0.9783796
## 639          skewness_yaw_dumbbell 13737 1.0000000
## 651              max_roll_dumbbell 13439 0.9783068
## 658             max_picth_dumbbell 13439 0.9783068
## 665               max_yaw_dumbbell 13441 0.9784524
## 672              min_roll_dumbbell 13439 0.9783068
## 679             min_pitch_dumbbell 13439 0.9783068
## 686               min_yaw_dumbbell 13441 0.9784524
## 693        amplitude_roll_dumbbell 13439 0.9783068
## 700       amplitude_pitch_dumbbell 13439 0.9783068
## 707         amplitude_yaw_dumbbell 13441 0.9784524
## 721             var_accel_dumbbell 13439 0.9783068
## 728              avg_roll_dumbbell 13439 0.9783068
## 735           stddev_roll_dumbbell 13439 0.9783068
## 742              var_roll_dumbbell 13439 0.9783068
## 749             avg_pitch_dumbbell 13439 0.9783068
## 756          stddev_pitch_dumbbell 13439 0.9783068
## 763             var_pitch_dumbbell 13439 0.9783068
## 770               avg_yaw_dumbbell 13439 0.9783068
## 777            stddev_yaw_dumbbell 13439 0.9783068
## 784               var_yaw_dumbbell 13439 0.9783068
## 875          kurtosis_roll_forearm 13505 0.9831113
## 882         kurtosis_picth_forearm 13506 0.9831841
## 884           kurtosis_yaw_forearm 13737 1.0000000
## 896          skewness_roll_forearm 13505 0.9831113
## 903         skewness_pitch_forearm 13506 0.9831841
## 905           skewness_yaw_forearm 13737 1.0000000
## 917               max_roll_forearm 13439 0.9783068
## 924              max_picth_forearm 13439 0.9783068
## 931                max_yaw_forearm 13505 0.9831113
## 938               min_roll_forearm 13439 0.9783068
## 945              min_pitch_forearm 13439 0.9783068
## 952                min_yaw_forearm 13505 0.9831113
## 959         amplitude_roll_forearm 13439 0.9783068
## 966        amplitude_pitch_forearm 13439 0.9783068
## 973          amplitude_yaw_forearm 13505 0.9831113
## 987              var_accel_forearm 13439 0.9783068
## 994               avg_roll_forearm 13439 0.9783068
## 1001           stddev_roll_forearm 13439 0.9783068
## 1008              var_roll_forearm 13439 0.9783068
## 1015             avg_pitch_forearm 13439 0.9783068
## 1022          stddev_pitch_forearm 13439 0.9783068
## 1029             var_pitch_forearm 13439 0.9783068
## 1036               avg_yaw_forearm 13439 0.9783068
## 1043            stddev_yaw_forearm 13439 0.9783068
## 1050               var_yaw_forearm 13439 0.9783068
```

```
exclude_vars=subset(train_summary1, PCNT_NA>0.5)$Var2

train2=train0[,-c(exclude_vars)]

test2=test0[,-c(exclude_vars)]
```

Step 5: Impute missing values and scale as well as mean-center the variables (in cases where it is not done):

```
prePro=preProcess(train2,method=c("knnImpute","center","scale"))
train3=predict(prePro,train2)
test3=predict(prePro,test2)
test=predict(prePro,test)
```

Step 6: Check for near zero variables

```
nsv=nearZeroVar(train3[,c(8:ncol(train3))],saveMetrics=TRUE)
nrow(subset(nsv, nzv==TRUE))
```

```
## [1] 0
```

Because non of the variables are near zero, non of them are excluded from the feature list at this step.

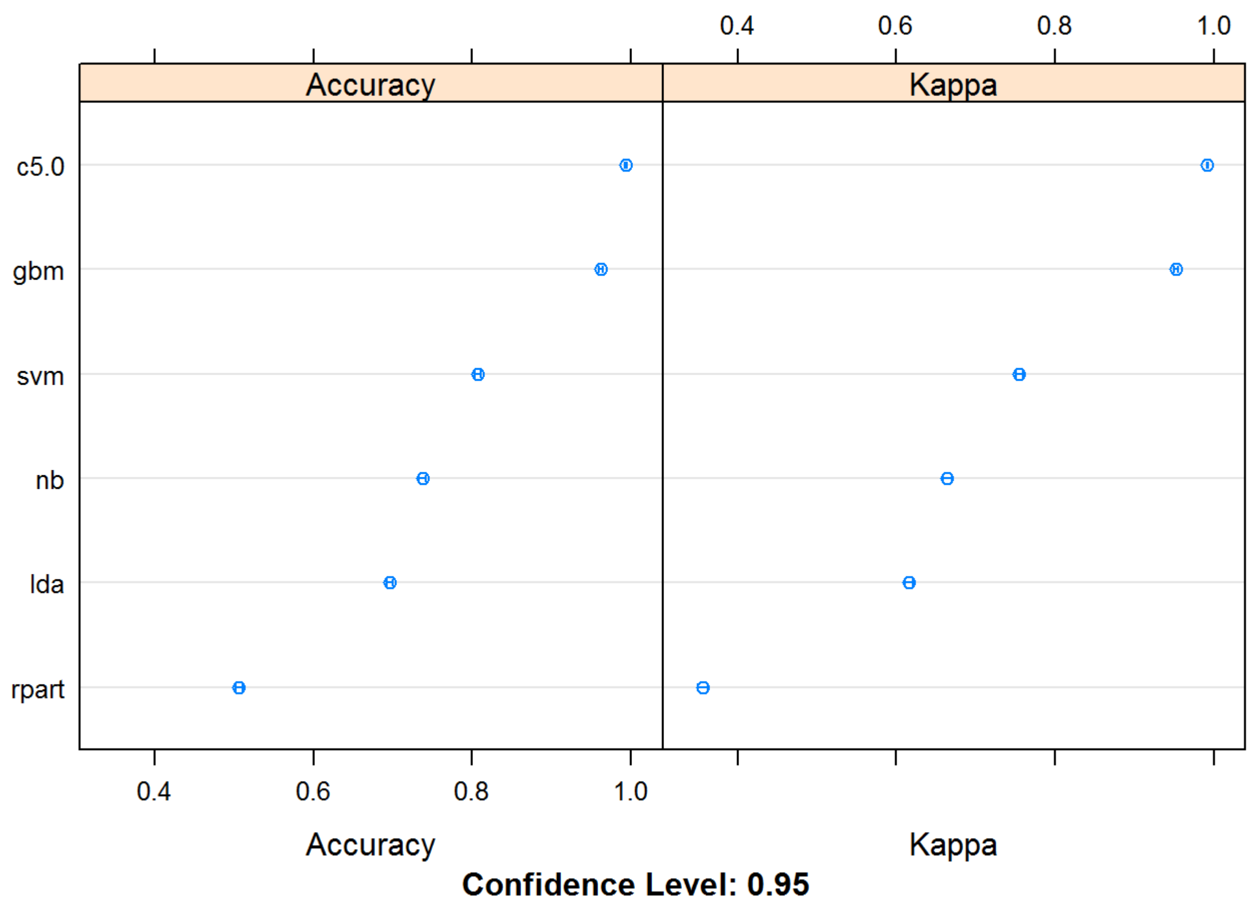
Model selection

Step 7: Run several different classification models using 3 repeats of 10-fold cross validation. List of models: 1) C5.0 2) Stochastic Gradient Boosting 3) Linear Discriminant Analysis 4) Support Vector Machine with a Radial Basis Kernel Function 5) Classification and Regression Trees

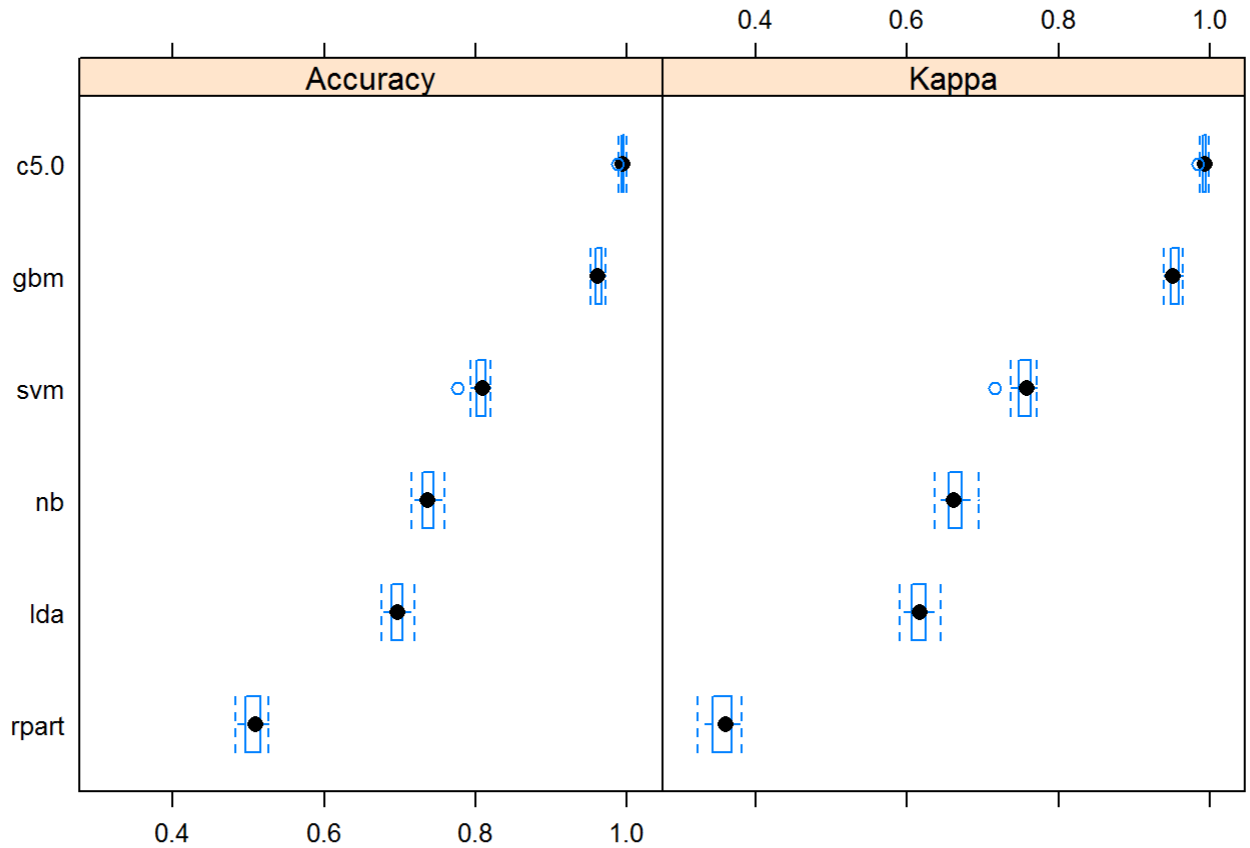
```
seed = 387
metric = "Accuracy"
control = trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(seed)
fit.c50 = train(classe~., data=train3[,c(8:ncol(train3))], method="C5.0", metric=metric, trContr
ol=control)
fit.gbm = train(classe~., data=train3[,c(8:ncol(train3))], method="gbm", metric=metric, trContro
l=control, verbose=FALSE)
fit.lda=train(classe~.,data=train3[,c(8:ncol(train3))], method="lda", metric=metric, trControl=c
ontrol)
fit.svm=train(classe~.,data=train3[,c(8:ncol(train3))], method="svmRadial", metric=metric, trCon
trol=control)
fit.rpart=train(classe~.,data=train3[,c(8:ncol(train3))], method="rpart", metric=metric, trContr
ol=control)

results = resamples(list(c5.0=fit.c50, gbm=fit.gbm, lda=fit.lda, svm=fit.svm, rpart=fit.rpart))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: c5.0, gbm, lda, svm, rpart
## Number of resamples: 30
##
## Accuracy
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## c5.0  0.9883636 0.9921721 0.9938161 0.9937401 0.9954465 0.9978166    0
## gbm   0.9519301 0.9601528 0.9643117 0.9629224 0.9655803 0.9766934    0
## lda   0.6790393 0.6894669 0.6969800 0.6970950 0.7043304 0.7210488    0
## svm   0.9081633 0.9177584 0.9200000 0.9219385 0.9269847 0.9366812    0
## rpart 0.4777859 0.4926268 0.5092764 0.5062006 0.5176550 0.5334789    0
##
## Kappa
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## c5.0  0.9852879 0.9900988 0.9921777 0.9920820 0.9942404 0.9972385    0
## gbm   0.9391691 0.9495797 0.9548384 0.9530897 0.9564434 0.9705137    0
## lda   0.5937144 0.6071955 0.6166332 0.6166544 0.6257136 0.6465821    0
## svm   0.8837681 0.8956587 0.8985949 0.9010738 0.9074413 0.9197691    0
## rpart 0.3165128 0.3364962 0.3581838 0.3549653 0.3700395 0.3929836    0
```

```
dotplot(results)
```



```
bwplot(results)
```



Step 8: Confirm that the model accuracy observed on the training data is consistent with the one on the validation data

```
confusionMatrix(test3$classe, predict(fit.c50,test3))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   11 1127    1    0    0
##          C    0   10 1011    5    0
##          D    0    1    2  959    2
##          E    0    0    0    0 1082
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9944          
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2862          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9929          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.9895   0.9970   0.9948   0.9982
## Specificity            0.9998   0.9975   0.9969   0.9990   1.0000
## Pos Pred Value         0.9994   0.9895   0.9854   0.9948   1.0000
## Neg Pred Value         0.9974   0.9975   0.9994   0.9990   0.9996
## Prevalence             0.2862   0.1935   0.1723   0.1638   0.1842
## Detection Rate         0.2843   0.1915   0.1718   0.1630   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9966   0.9935   0.9970   0.9969   0.9991
```

```
confusionMatrix(test3$classe, predict(fit.gbm,test3))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1653   15    4    0    2
##          B   39 1071   25    4    0
##          C    0   31  978   14    3
##          D    2    3   26  923   10
##          E    0   15    5   15 1047
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9638          
##                  95% CI : (0.9587, 0.9684)
##     No Information Rate : 0.2879          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9542          
##  Mcnemar's Test P-Value : 2.031e-05       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9758   0.9436   0.9422   0.9655   0.9859
## Specificity            0.9950   0.9857   0.9901   0.9917   0.9927
## Pos Pred Value         0.9875   0.9403   0.9532   0.9575   0.9677
## Neg Pred Value         0.9903   0.9865   0.9877   0.9933   0.9969
## Prevalence             0.2879   0.1929   0.1764   0.1624   0.1805
## Detection Rate         0.2809   0.1820   0.1662   0.1568   0.1779
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9854   0.9646   0.9661   0.9786   0.9893
```

```
confusionMatrix(test3$classe, predict(fit.lda,test3))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1371   36  139  125    3
##          B  182  738  128   43   48
##          C   99   89  666  140   32
##          D   56   42  106  722   38
##          E   34  171   96  109  672
##
## Overall Statistics
##
##                Accuracy : 0.7084
##                  95% CI : (0.6966, 0.72)
##     No Information Rate : 0.296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.631
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7870   0.6859   0.5868   0.6339   0.8474
## Specificity            0.9269   0.9166   0.9242   0.9490   0.9195
## Pos Pred Value         0.8190   0.6479   0.6491   0.7490   0.6211
## Neg Pred Value         0.9119   0.9288   0.9035   0.9153   0.9748
## Prevalence             0.2960   0.1828   0.1929   0.1935   0.1347
## Detection Rate         0.2330   0.1254   0.1132   0.1227   0.1142
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8569   0.8012   0.7555   0.7914   0.8834
```

```
confusionMatrix(test3$classe, predict(fit.svm,test3))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1660    8    5    0    1
##          B  104  996   35    1    3
##          C    4   42  958   17    5
##          D   11    2   88  863    0
##          E    3   13   37   29 1000
##
## Overall Statistics
##
##                Accuracy : 0.9307
##                  95% CI : (0.9239, 0.937)
##     No Information Rate : 0.3028
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9121
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9315   0.9387   0.8531   0.9484   0.9911
## Specificity            0.9966   0.9704   0.9857   0.9797   0.9832
## Pos Pred Value         0.9916   0.8745   0.9337   0.8952   0.9242
## Neg Pred Value         0.9710   0.9863   0.9660   0.9904   0.9981
## Prevalence             0.3028   0.1803   0.1908   0.1546   0.1715
## Detection Rate         0.2821   0.1692   0.1628   0.1466   0.1699
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9641   0.9545   0.9194   0.9640   0.9871
```

```
confusionMatrix(test3$classe, predict(fit.rpart,test3))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1515   31  124    0    4
##          B  468  392  279    0    0
##          C  457   29  540    0    0
##          D  437  185  342    0    0
##          E  158  148  271    0  505
## 
## Overall Statistics
##                                           
##                Accuracy : 0.5016          
##                  95% CI : (0.4888, 0.5145)
##     No Information Rate : 0.5157          
##     P-Value [Acc > NIR] : 0.9853          
##                                           
##                   Kappa : 0.3489          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4992  0.49936  0.34704       NA  0.99214
## Specificity            0.9442  0.85353  0.88773   0.8362  0.89267
## Pos Pred Value         0.9050  0.34416  0.52632       NA  0.46673
## Neg Pred Value         0.6390  0.91719  0.79090       NA  0.99917
## Prevalence             0.5157  0.13339  0.26440   0.0000  0.08649
## Detection Rate         0.2574  0.06661  0.09176   0.0000  0.08581
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7217  0.67645  0.61739       NA  0.94241
```

Results

As illustrated in the above figures and results, C5.0 algirithm showed best performance at predicting the type of excersize based on the available features. Thus it was chosen as the final model. C5.0 model consufion matrix illustrates the out of sample error.

Step 9: Predicting the types of excersizes for the 20 cases in the test set

```
predict(fit.c50,test)
```

```
##  [1] 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
```