Weight Lifting Project Code

Jennifre Richards

Friday, March 20, 2015

IN THE COMPILED HTML I reduced the number of trees in the random forest from 1000 to 10, because it took several hours to run the 1000 trees I used to build my random forest models; thus the results in this document are different from the results presented in the project write-up.

```
setwd('C:/Rrepos/Rdata/PML')
getwd()
```

```
## [1] "C:/Rrepos/Rdata/PML"
```

```
#install.packages('caret', dependencies=c('Depends', 'Suggests'))
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# IN THE COMPILED HTML I reduced the number of trees in the random forest from
# 1000 to 10, because it took several hours to run the 1000 trees; thus the results
# in this document are different from the results presented in the project write-up.

# loading the data
test <- read.csv('C:/Rrepos/Rdata/PML/testing.csv', stringsAsFactors=T, header = T)
train <- read.csv('C:/Rrepos/Rdata/PML/training.csv', stringsAsFactors=T, header = T)

# preparing the data;since goal was to predict, eliminated data that did not seem relevan
t (some of initial
# columns, including individual names, and then data that had no values or NAs in the tes
ting dataset; this corresponded to similar but not
# completely equivalent data in the training set (i.e., were some values for kurtosis, et
c.)); note that Adelmo
# had "0" for roll, pitch and yaw forearm

train1 <- train[,-c(1, 3:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150)]
test1 <- test[,-c(1, 3:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150)]
train1 <- train1[,c(2:53,1,54)]
test1 <- test1[,c(2:53,1,54)]

#-------------------------------------------------------------
# subsetting the data into training and test data
inTrain <- createDataPartition(y=train1$class, p=0.7, list=FALSE)

training <- train1[inTrain,]
testing <- train1[inTrain,]

#looking for complete cases
train2 <- complete.cases(train1)
train3 <-train1[train2,]
nrow(train3)
```

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

```
# look at distribution of samples among classes in all three data sets
prop.orig <- prop.table(table(train1[54]))
original_Proportions <- prop.orig
prop.train <- prop.table(table(training[54]))
training_Proportions <- prop.train
prop.test <- prop.table(table(testing[54]))
testing_Proportions <- prop.test
Table1 <- rbind(original_Proportions,training_Proportions,testing_Proportions)
Table1
```

```
##                               A         B         C         D         E
## original_Proportions 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
## training_Proportions 0.2843416 0.1934920 0.1744195 0.1639368 0.1838101
## testing_Proportions  0.2843416 0.1934920 0.1744195 0.1639368 0.1838101
```

```
# check for near zero varaiance predictors
training1 <- nearZeroVar(training)
training1
```

```
## integer(0)
```

```
# check for highly correlated variables and remove
Corr <- cor(training[-c(53:54)])
highCorr <- findCorrelation(Corr, 0.90)
names(training[highCorr])
```

```
## [1] "accel_belt_z" "roll_belt"    "accel_belt_y" "accel_belt_x"
## [5] "gyros_arm_y"
```

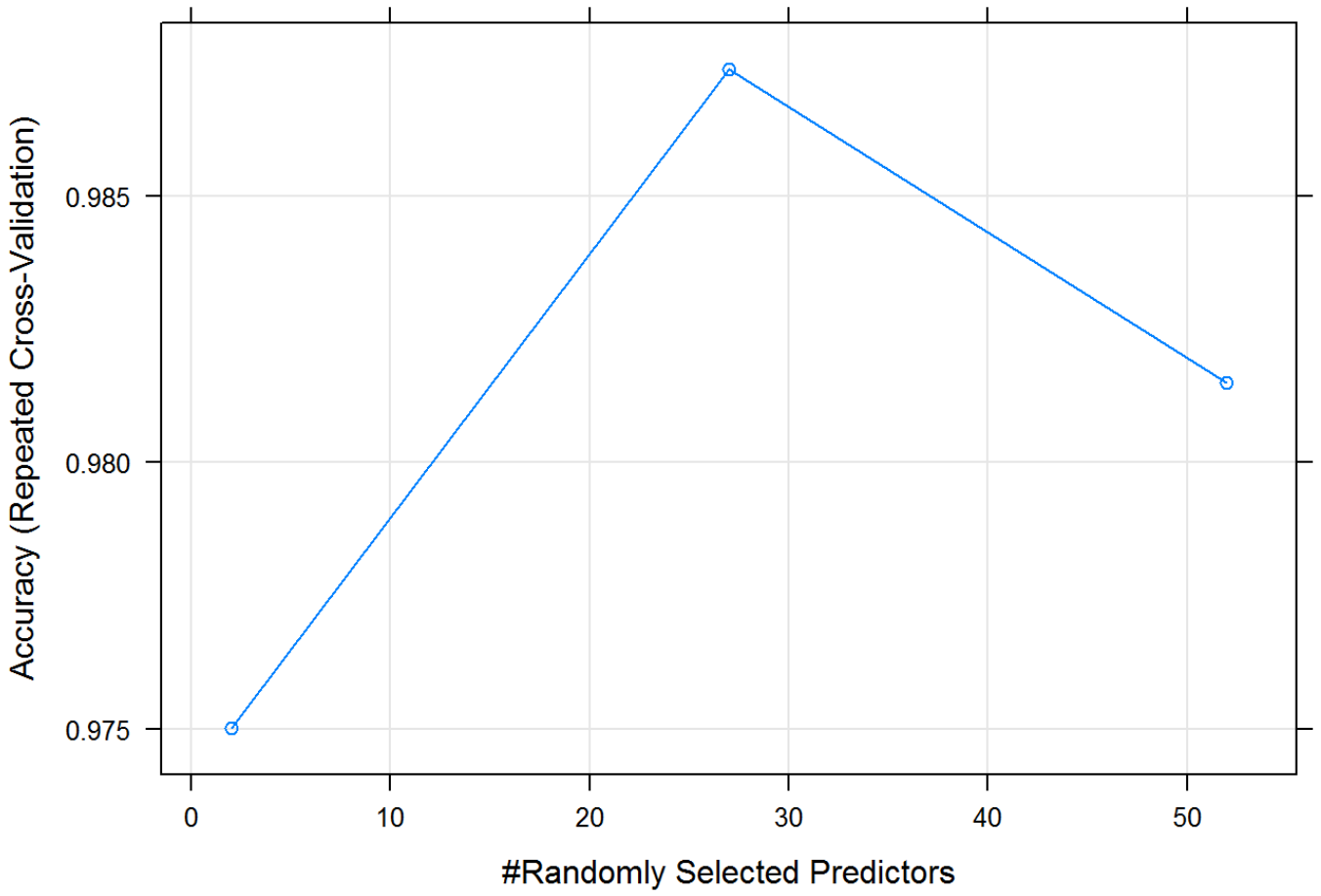
```
training <- training[,-highCorr]
testing <- testing[,-highCorr]
test2 <- test1[,-highCorr]

#------------------------------------------------------------
# fitting a model and detemining variable importance using random forest
cvCtrl <- trainControl(method='repeatedcv', repeats=3, classProbs=TRUE)
modFit <- train(class~., data=training, method='rf', ntree=10, trControl=cvCtrl)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

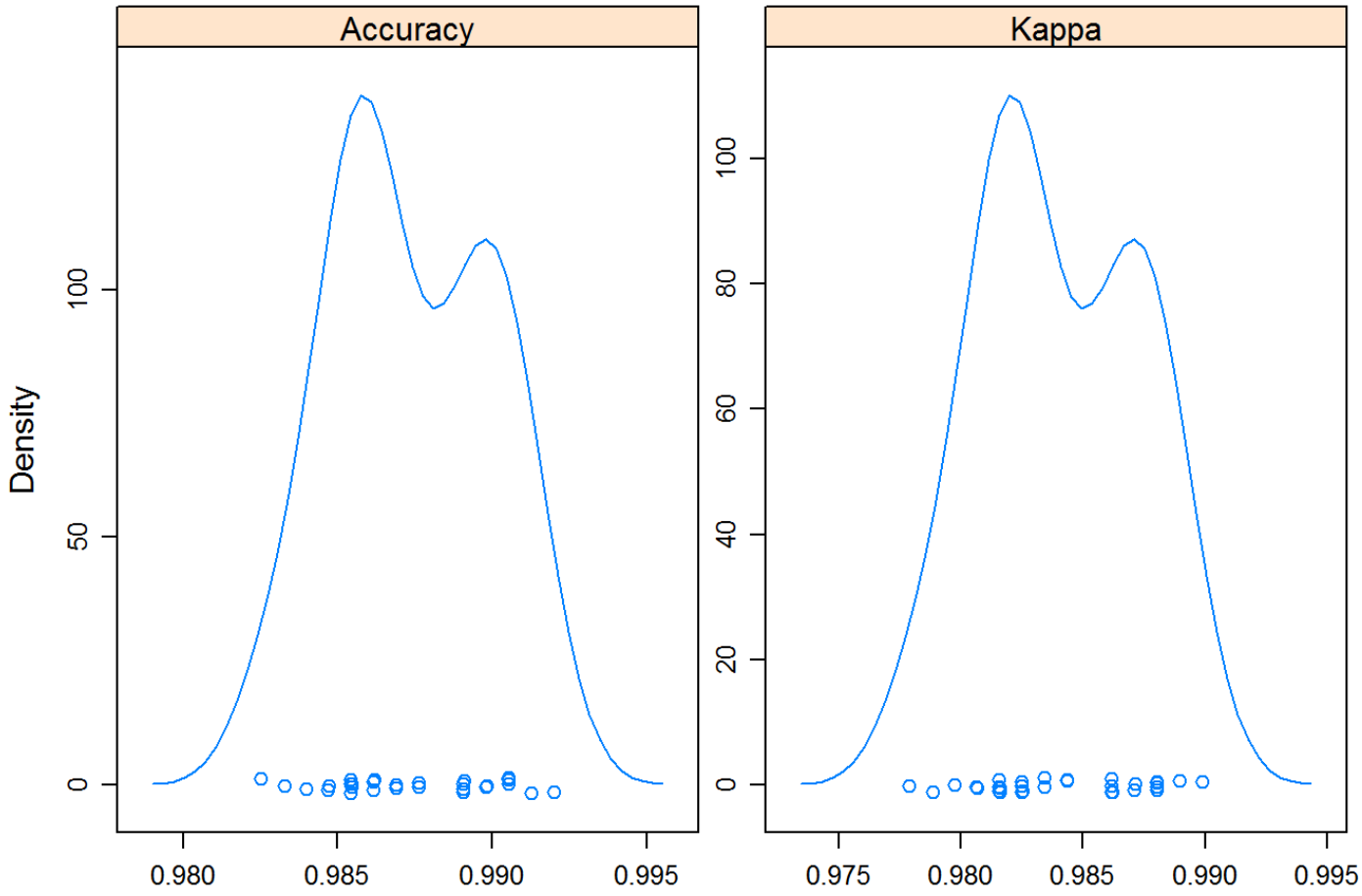
```
modFit
```

```
## Random Forest
##
## 13737 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
##
## Summary of sample sizes: 12364, 12362, 12362, 12365, 12362, 12364, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9750073  0.9683752  0.004070593  0.005147960
##   27    0.9873821  0.9840385  0.002566086  0.003246380
##   52    0.9814852  0.9765747  0.003921958  0.004965011
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
#looking at plots of results and variable importance
Y <- plot(modFit)
Y
```




```
p <- resampleHist(modFit)
p
```




```
Imp <- varImp(modFit)
Imp
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## yaw_belt              100.00
## pitch_forearm          84.68
## pitch_belt             66.44
## magnet_dumbbell_z      64.38
## magnet_dumbbell_y      46.22
## roll_forearm           37.50
## accel_dumbbell_y       31.67
## magnet_belt_z          28.72
## roll_dumbbell          28.05
## magnet_belt_y          27.26
## total_accel_belt       25.05
## gyros_belt_z           23.81
## total_accel_dumbbell   23.26
## accel_dumbbell_z       20.22
## magnet_dumbbell_x      18.13
## yaw_dumbbell           17.84
## accel_forearm_z        17.42
## magnet_forearm_z       17.41
## magnet_belt_x          17.21
## accel_forearm_x        17.03
```

```
# getting column numbers for important features
which(colnames(training)=='yaw_belt')
```

```
## [1] 2
```

```
which(colnames(training)=='pitch_forearm')
```

```
## [1] 36
```

```
which(colnames(training)=='pitch_belt')
```

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

```
which(colnames(training)=='magnet_dumbbell_z')
```

```
## [1] 34
```

```
which(colnames(training)=='magnet_dumbbell_y')
```

```
## [1] 33
```

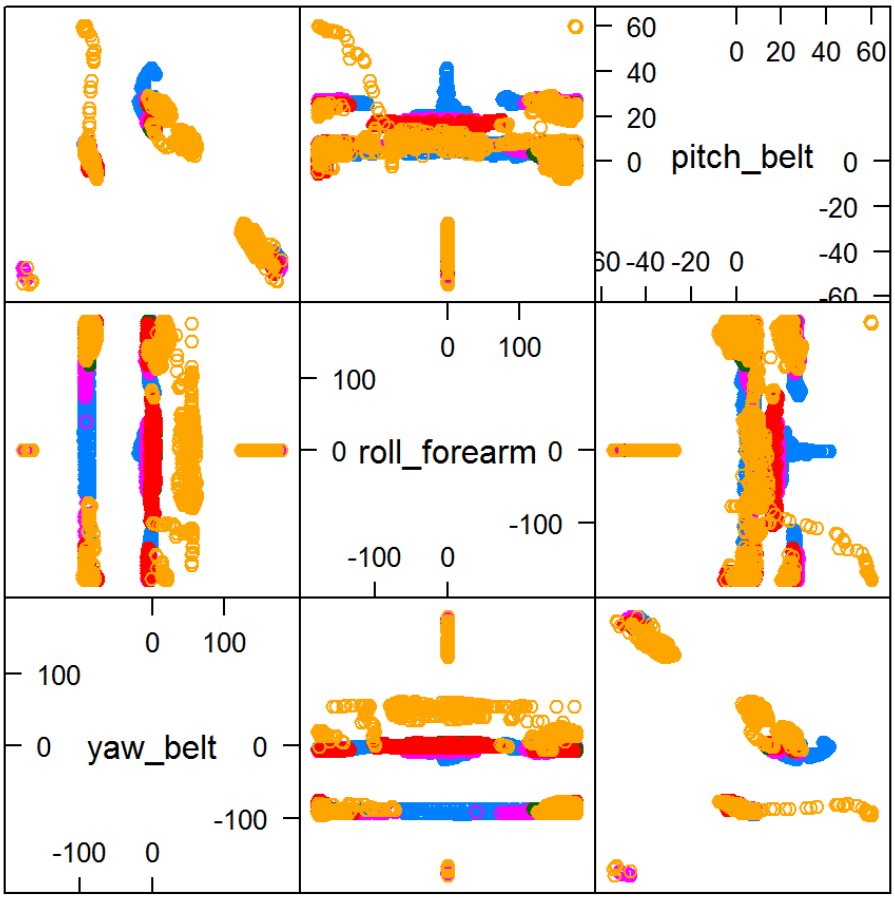
```
which(colnames(training)=='roll_forearm')
```

```
## [1] 35
```

```
which(colnames(training)=='magnet_belt_y')
```

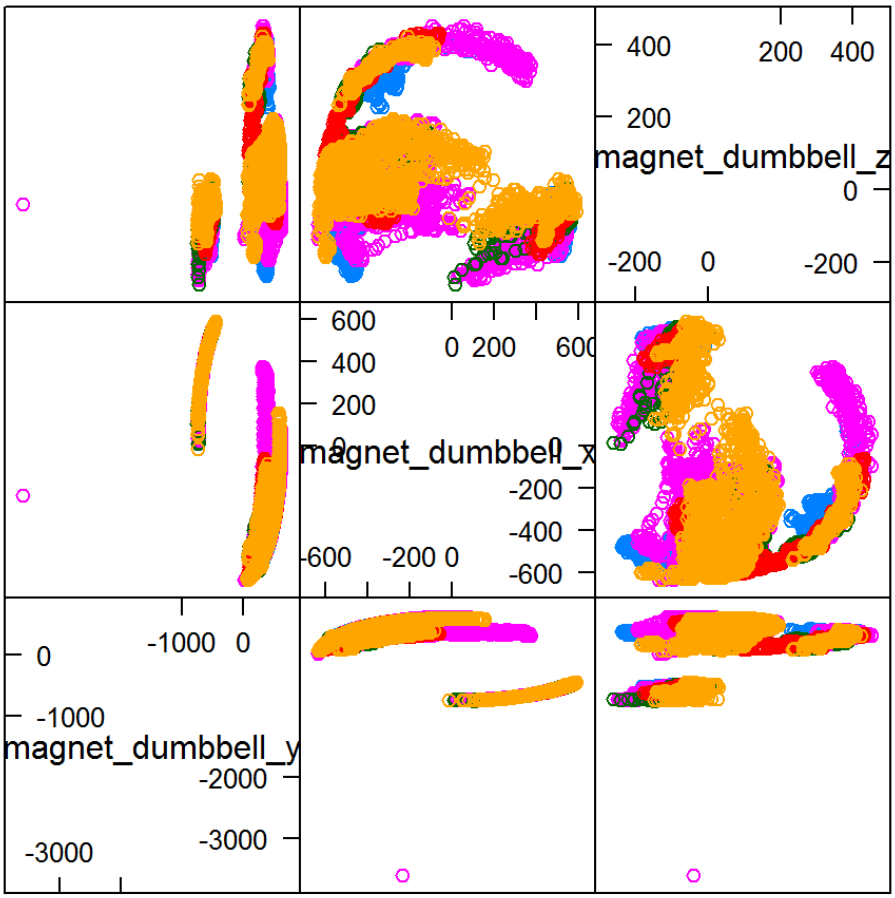
```
## [1] 8
```

```
#-----------------------------------------------------------------
# plotting

# plotting 4 most important features against each other
featurePlot(x=training[,c(2,35,1)],
            y=training$class,
            plot='pairs')
```



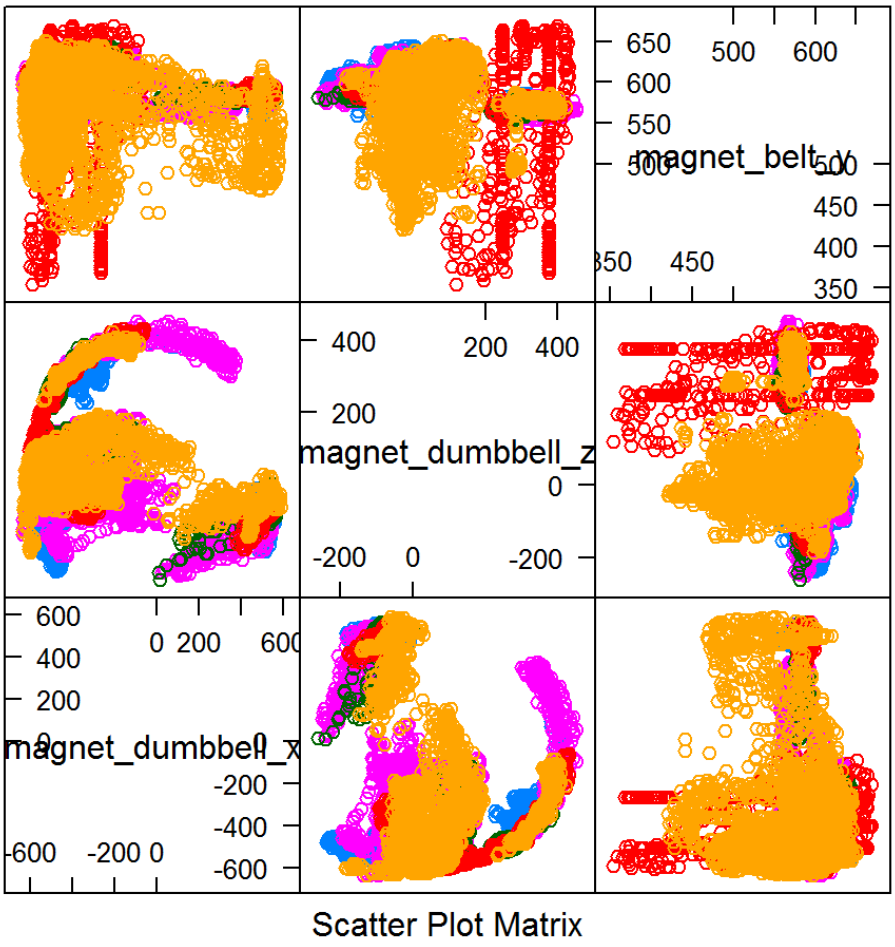
Scatter Plot Matrix

```
featurePlot(x=training[,c(33,32,34)],
            y=training$class,
            plot='pairs')
```

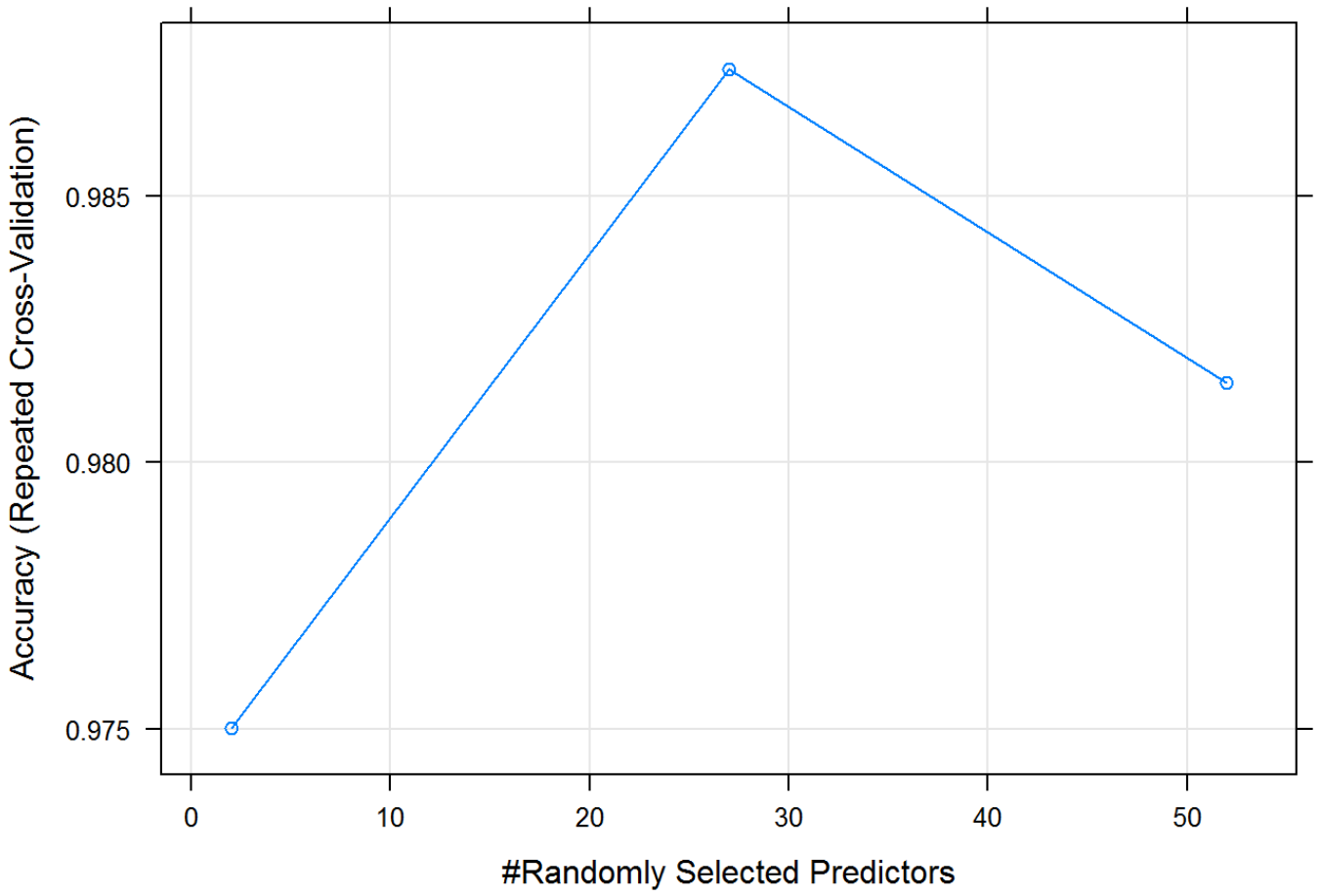


Scatter Plot Matrix

```
featurePlot(x=training[,c(32,34,8)],
            y=training$class,
            plot='pairs')
```




```
# plotting number of predictors vs. accuracy
plot(modFit)
```




```
#-------------------------------------------------------------------------
# predict weight lifing class using model and getting confusion matrix
predWL <- predict(modFit, newdata=testing)
str(predWL)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
confMatrix1 <- confusionMatrix(data=predWL, testing$class)
confMatrix1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    2    0
##          D    0    0    0 2250    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 0.9999
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9998
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   0.9991   1.0000
## Specificity            1.0000   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9992   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9998   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1638   0.1838
## Detection Prevalence   0.2843   0.1935   0.1746   0.1638   0.1838
## Balanced Accuracy      1.0000   1.0000   0.9999   0.9996   1.0000
```

```
#----------------------------------------------------------------------
#predicting testing data
pred <- predict(modFit,test2)
TestPred <- as.character(pred)
TestPred
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```
#-----------------------------------------------------------------
# repeating the model building with the 7 most important variables

# loading the data
test <- read.csv('C:/Rrepos/Rdata/PML/testing.csv', stringsAsFactors=T, header = T)
train <- read.csv('C:/Rrepos/Rdata/PML/training.csv', stringsAsFactors=T, header = T)

# preparing the data;since goal was to predict, eliminated data that did not seem relevan
t
# (some of initialcolumns, including individual names, and then data that had no values o
r
# NAs in the testing dataset; this corresponded to similar but not completely equivalent
# data in the training set (i.e., were some values for kurtosis, etc.)); note that Adelmo
# had "0" for roll, pitch and yaw forearm

train1 <- train[,-c(1, 3:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150)]
test1 <- test[,-c(1, 3:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150)]

#-------------------------------------------------------------
# subsetting the data into training and test data
inTrain <- createDataPartition(y=train1$class, p=0.7, list=FALSE)

training <- train1[inTrain,]
testing <- train1[-inTrain,]

#looking for complete cases
train2 <- complete.cases(train1)
train3 <-train1[train2,]
nrow(train3)
```

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

```
which(colnames(training)=='yaw_belt')
```

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

```
which(colnames(training)=='pitch_forearm')
```

```
## [1] 42
```

```
which(colnames(training)=='pitch_belt')
```

```
## [1] 3
```

```
which(colnames(training)=='magnet_dumbbell_z')
```

```
## [1] 40
```

```
which(colnames(training)=='magnet_dumbbell_y')
```

```
## [1] 39
```

```
which(colnames(training)=='roll_forearm')
```

```
## [1] 41
```

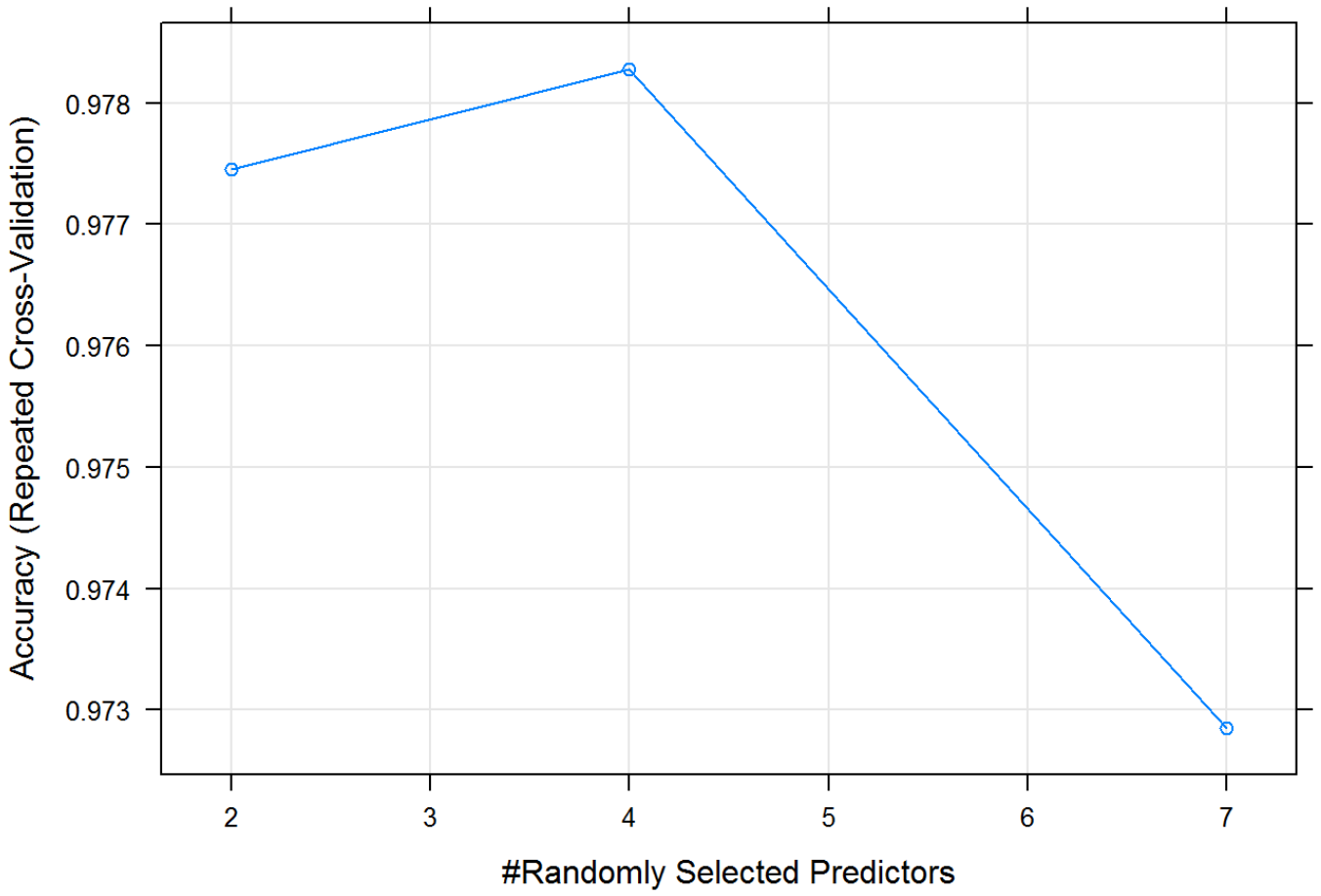
```
which(colnames(training)=='magnet_belt_y')
```

```
## [1] 13
```

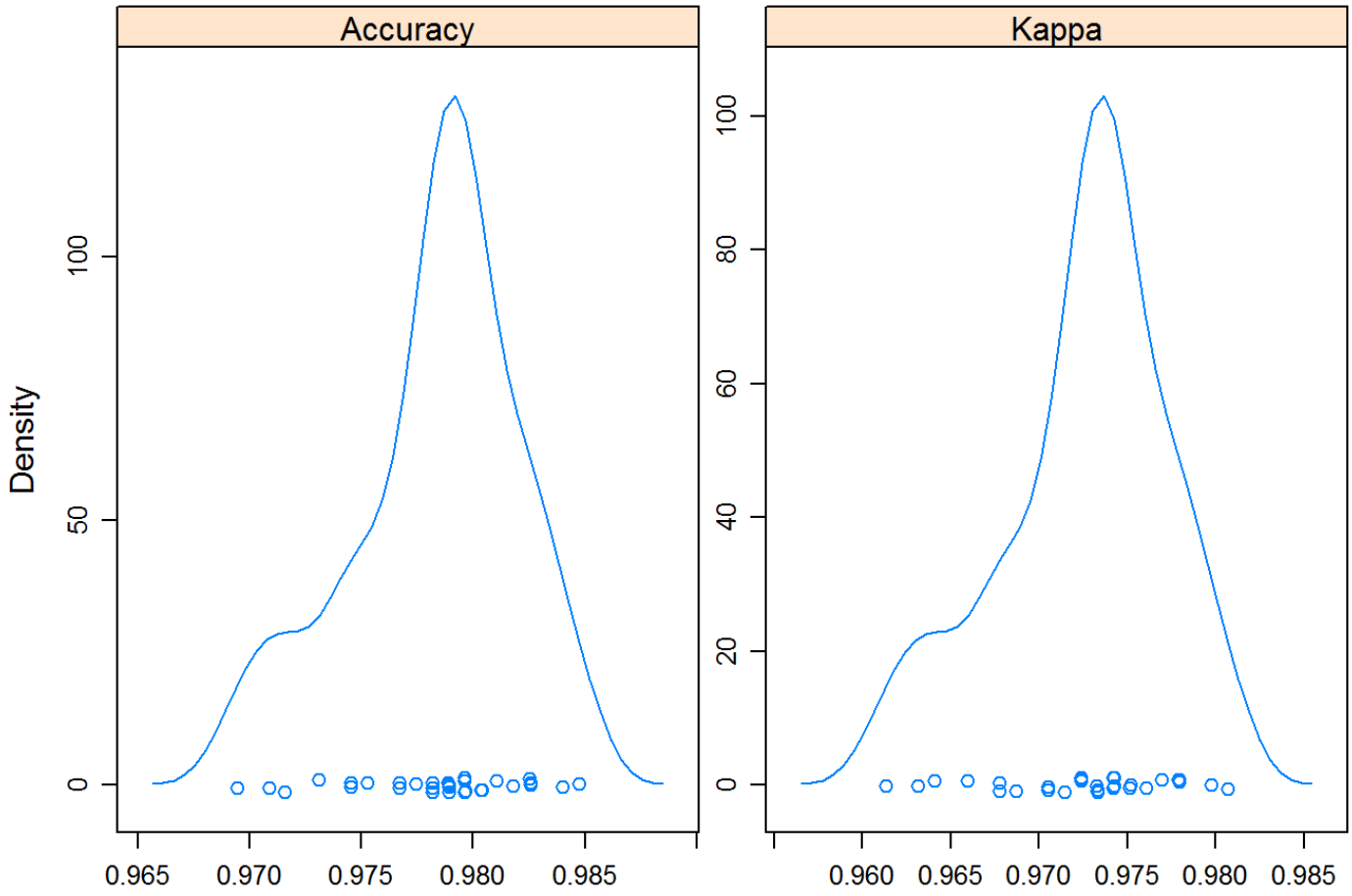
```
# creating subsets of data that had just the 7 most important variables
trainSub <- training[,c(3:4, 13, 39:42, 54)]
testSub <- testing[,c(3:4, 13, 39:42, 54)]
test1Sub <- test1[,c(3:4, 13, 39:42, 54)]

#------------------------------------------------------------
# fitting a model and detemining variable importance using random forest
cvCtrl <- trainControl(method='repeatedcv', repeats=3, classProbs=TRUE)
modFit <- train(class~., data=trainSub, method='rf', ntree=100, trControl=cvCtrl)
modFit
```

```
## Random Forest
##
## 13737 samples
##     7 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
##
## Summary of sample sizes: 12365, 12362, 12363, 12363, 12363, 12363, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##   2     0.9774570  0.9714953  0.004095824  0.005176381
##   4     0.9782816  0.9725346  0.003780948  0.004780471
##   7     0.9728469  0.9656621  0.003733484  0.004719376
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 4.
```

```
Y <- plot(modFit)
Y
```




```
p <- resampleHist(modFit)
p
```




```
Imp <- varImp(modFit)
Imp
```

```
## rf variable importance
##
##                   Overall
## yaw_belt           100.00
## pitch_belt          52.39
## pitch_forearm       30.20
## magnet_dumbbell_z   25.96
## magnet_dumbbell_y   14.10
## magnet_belt_y        3.95
## roll_forearm         0.00
```

```
# getting column numbers for important features
which(colnames(trainSub)=='yaw_belt')
```

```
## [1] 2
```

```
which(colnames(trainSub)=='pitch_forearm')
```

```
## [1] 7
```

```
which(colnames(trainSub)=='pitch_belt')
```

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

```
which(colnames(trainSub)=='magnet_dumbbell_z')
```

```
## [1] 5
```

```
which(colnames(trainSub)=='magnet_dumbbell_y')
```

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

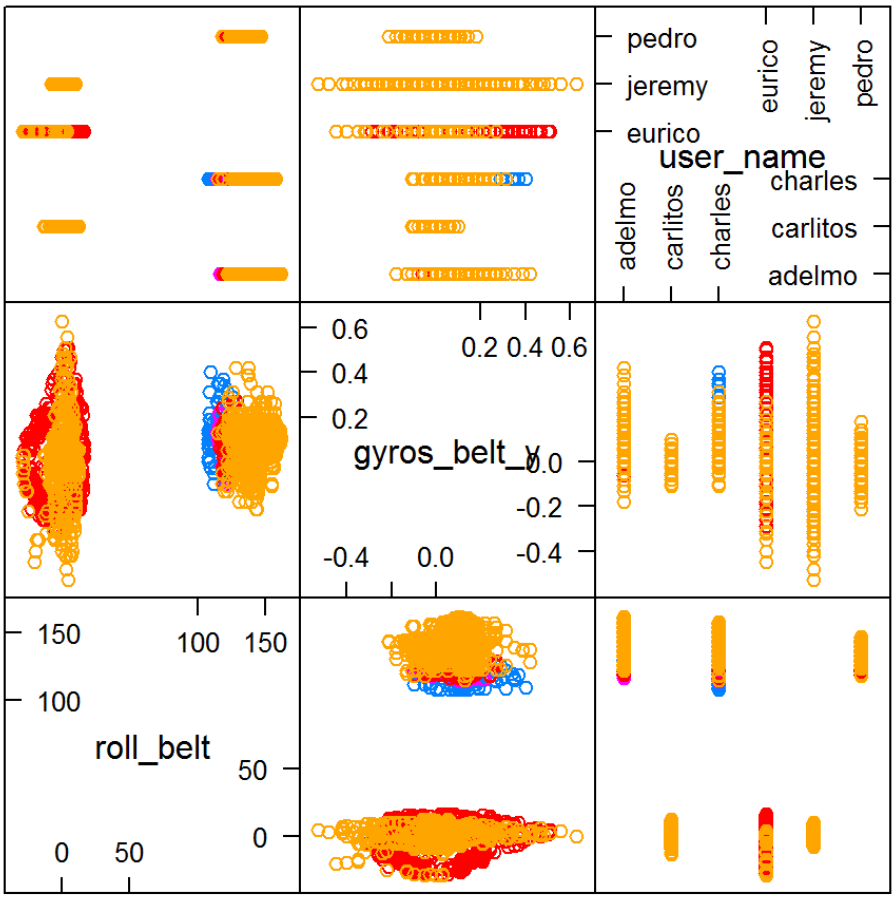
```
which(colnames(trainSub)=='roll_forearm')
```

```
## [1] 6
```

```
which(colnames(trainSub)=='magnet_belt_y')
```

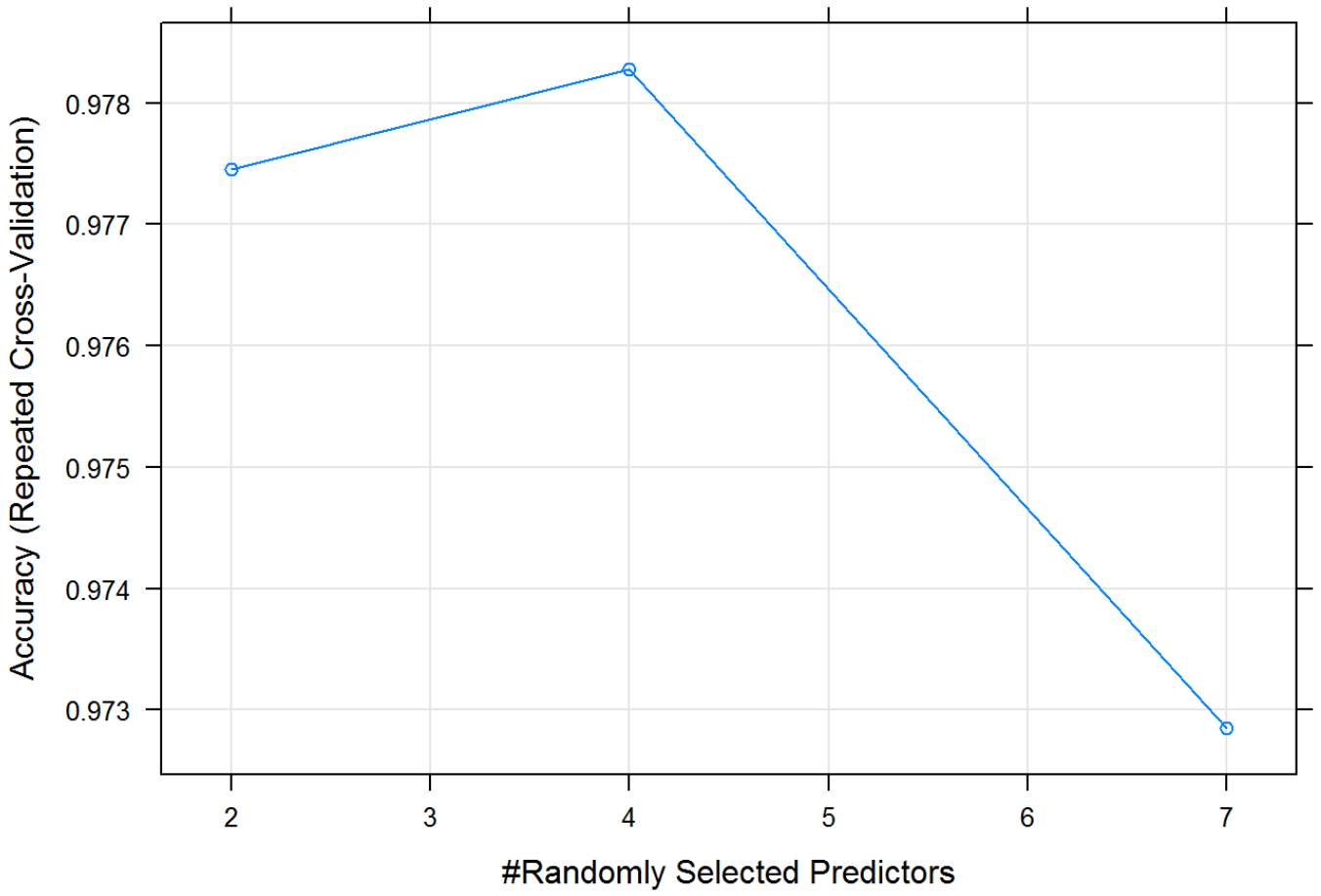
```
## [1] 3
```

```
#---------------------------------------------------------------------
# plotting

# plotting 4 most important features against each other
featurePlot(x=training[,c(2,7,1)],
            y=training$class,
            plot='pairs')
```



Scatter Plot Matrix

```
# plotting number of predictors vs. accuracy
plot(modFit)
```




```
#------------------------------------------------------------------------
# predict weight lifing class from testing data set using model and getting confusion mat
rix
predWL <- predict(modFit, newdata=testSub)

confMatrix1 <- confusionMatrix(data=predWL, testSub$class)
confMatrix1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1653   13    5    0    0
##          B   14 1101   13    0    9
##          C    6   17 1000    9    5
##          D    1    8    8  952    5
##          E    0    0    0    3 1063
##
## Overall Statistics
##
##                Accuracy : 0.9803
##                  95% CI : (0.9764, 0.9837)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9751
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9875   0.9666   0.9747   0.9876   0.9824
## Specificity            0.9957   0.9924   0.9924   0.9955   0.9994
## Pos Pred Value         0.9892   0.9683   0.9643   0.9774   0.9972
## Neg Pred Value         0.9950   0.9920   0.9946   0.9976   0.9961
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2809   0.1871   0.1699   0.1618   0.1806
## Detection Prevalence   0.2839   0.1932   0.1762   0.1655   0.1811
## Balanced Accuracy      0.9916   0.9795   0.9835   0.9915   0.9909
```

```
#---------------------------------------------------------------------
#predicting testing data
pred <- predict(modFit,test1Sub)
TestPred <- as.character(pred)
TestPred
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```