Discovery_dataset

Ariel-ac4391

11/22/2018

```r
training_data=read.csv("data/Data_User_Modeling_training_Dataset.csv")
test_data=read.csv("data/Data_User_Modeling_test_Dataset.csv")
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(ggplot2)
library(partykit)
```

```
## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm
```

```r
library(rpart) # Popular decision tree algorithm
library(hier.part)
```

```
## Loading required package: gtools
```

```r
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
```

```r
library(ipred)
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

```r
#library(rattle) # GUI for building trees and fancy tree plot #Doesn't work
library(rpart.plot) # Enhanced tree plots
library(party) # Alternative decision tree algorithm
```

```
## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

##
## Attaching package: 'party'

## The following objects are masked from 'package:partykit':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```r
library(partykit) # Convert rpart object to BinaryTree
#library(RWeka) # Weka decision tree J48.
library(C50) # Original C5.0 implementation.
library(e1071) # naive bayes
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:gtools':
##
##     permutations
```

```r
library(DMwR) # KNN
```

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

```r
summary(training_data)
```

```
##       STG              SCG              STR              LPR        
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.2407   1st Qu.:0.2100   1st Qu.:0.2913   1st Qu.:0.2500  
##  Median :0.3270   Median :0.3025   Median :0.4900   Median :0.3300  
##  Mean   :0.3711   Mean   :0.3557   Mean   :0.4680   Mean   :0.4327  
##  3rd Qu.:0.4950   3rd Qu.:0.4975   3rd Qu.:0.6900   3rd Qu.:0.6475  
##  Max.   :0.9900   Max.   :0.9000   Max.   :0.9500   Max.   :0.9900  
##       PEG               UNS    
##  Min.   :0.0000   High    :63  
##  1st Qu.:0.2500   Low     :83  
##  Median :0.5000   Middle  :88  
##  Mean   :0.4585   very_low:24  
```

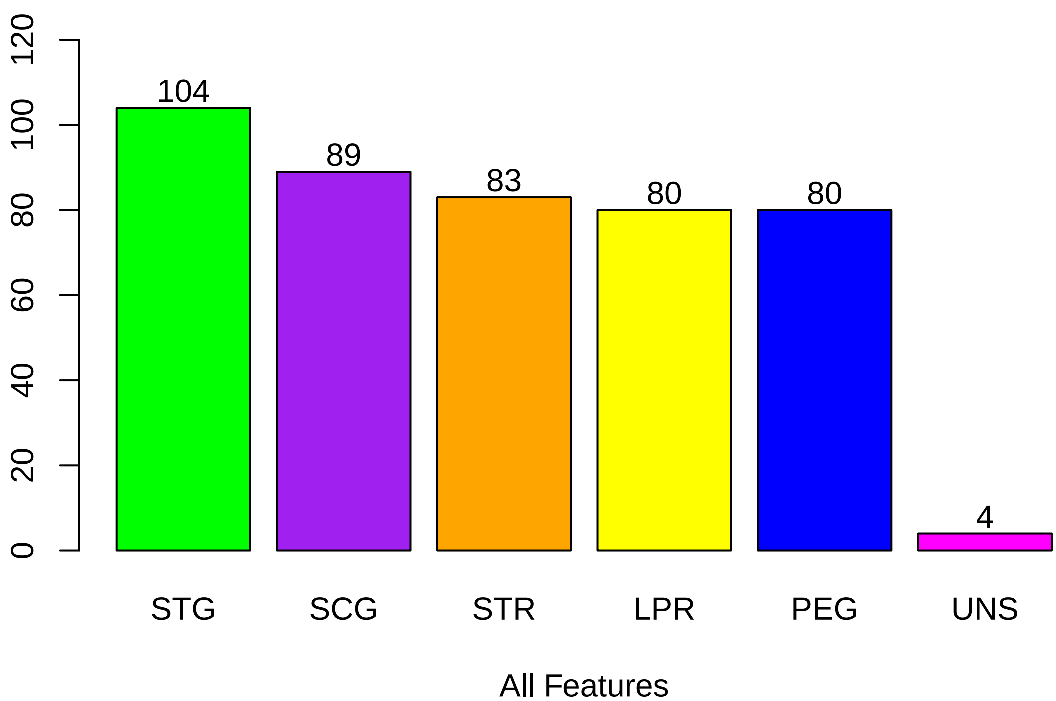
```
##   3rd Qu.:0.6600
##   Max.   :0.9300
```

```r
attach(training_data)
```

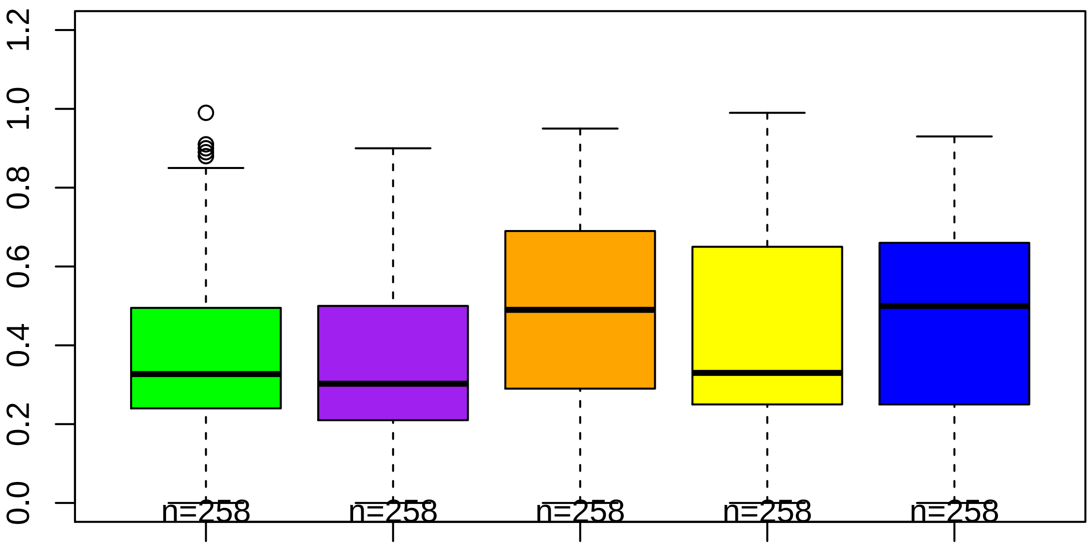
```r
summary(training_data)
```

```
##       STG              SCG              STR              LPR
##   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##   1st Qu.:0.2407   1st Qu.:0.2100   1st Qu.:0.2913   1st Qu.:0.2500
##   Median :0.3270   Median :0.3025   Median :0.4900   Median :0.3300
##   Mean   :0.3711   Mean   :0.3557   Mean   :0.4680   Mean   :0.4327
##   3rd Qu.:0.4950   3rd Qu.:0.4975   3rd Qu.:0.6900   3rd Qu.:0.6475
##   Max.   :0.9900   Max.   :0.9000   Max.   :0.9500   Max.   :0.9900
##       PEG               UNS
##   Min.   :0.0000   High    :63
##   1st Qu.:0.2500   Low     :83
##   Median :0.5000   Middle  :88
##   Mean   :0.4585   very_low:24
##   3rd Qu.:0.6600
##   Max.   :0.9300
```

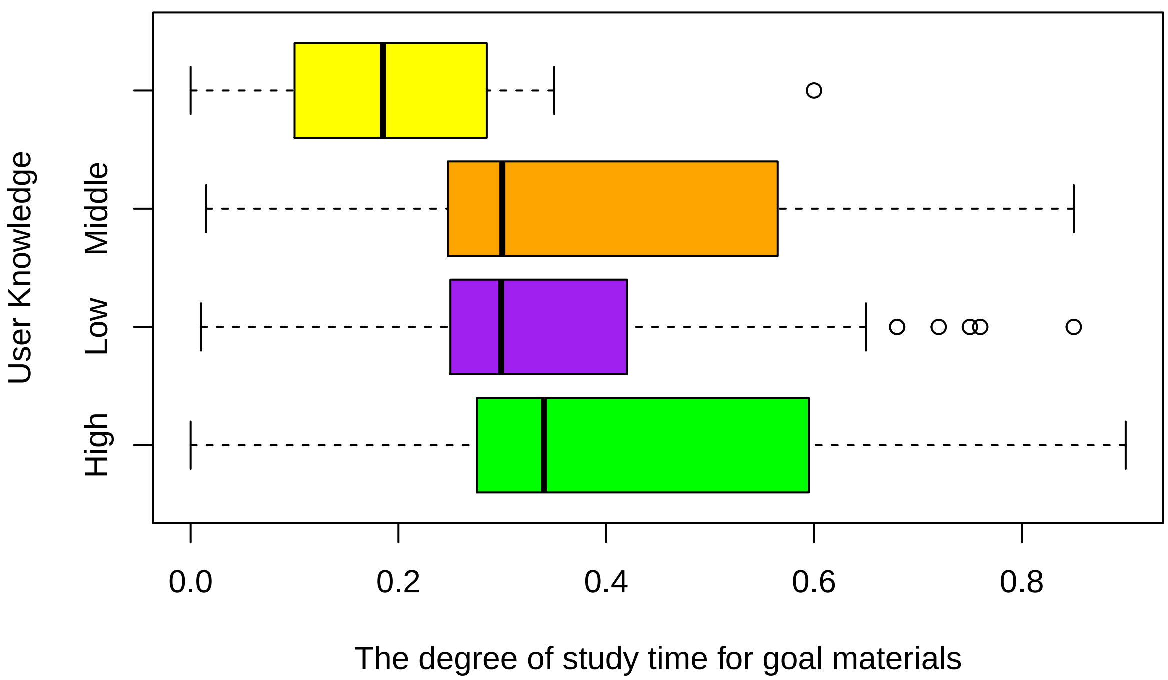
```r
# Number of distinct values in each feture
a = n_distinct(STG)
b = n_distinct(SCG)
c = n_distinct(STR)
d = n_distinct(LPR)
e = n_distinct(PEG)
f = n_distinct(UNS)
num_distinct = c(a,b,c,d,e,f)

plot = barplot(num_distinct, names = c("STG", "SCG", "STR", "LPR", "PEG", "UNS"), ylim=c(0,120), xlab="
text(plot,num_distinct + 4,labels=as.character(num_distinct))
```



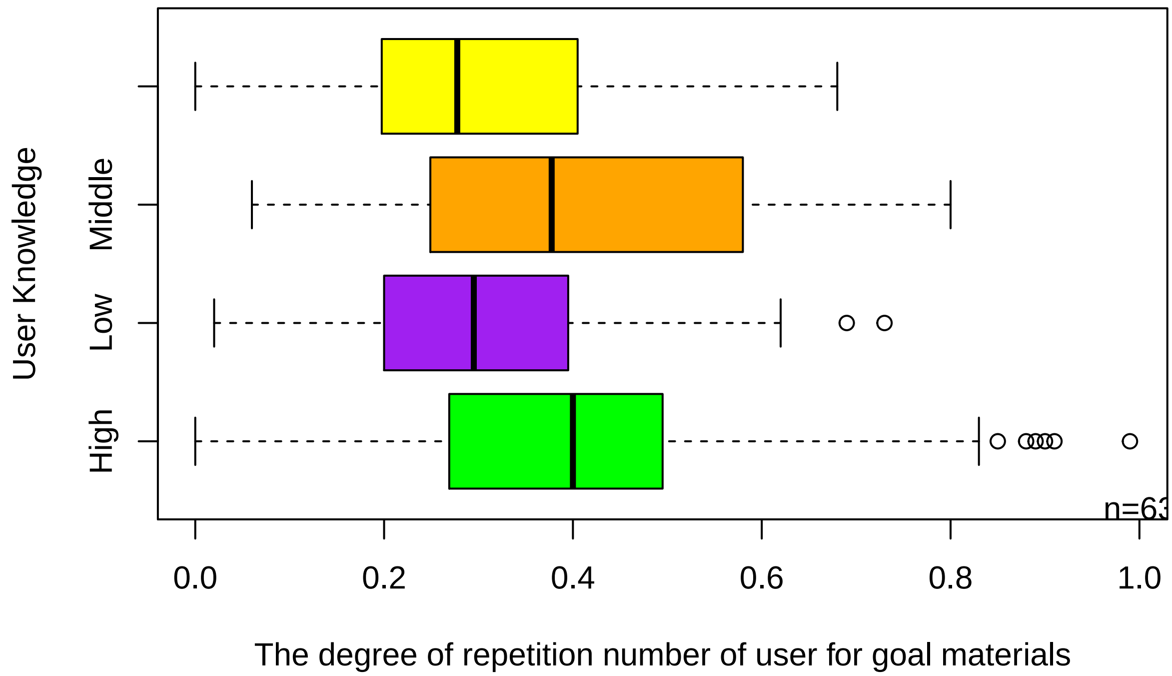
```
# boxplot of all data
boxplot2(STG,SCG,STR,LPR,PEG, col=c("green", "purple", "orange", "yellow", "blue", "magenta"), ylim=c(0
```



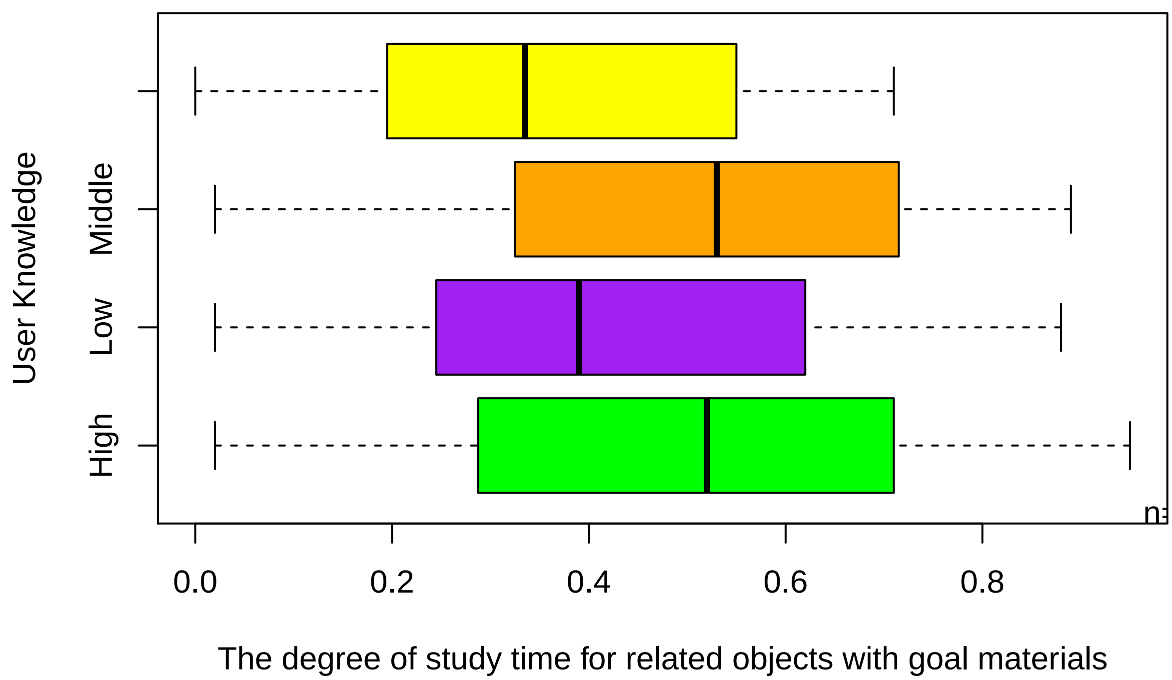

```
# boxplot of SCG divided accross UNS values
boxplot2(SCG~UNS,data=training_data, horizontal = TRUE,
    xlab="The degree of study time for goal materials", ylab="User Knowledge", col=c("green", "purple"
```



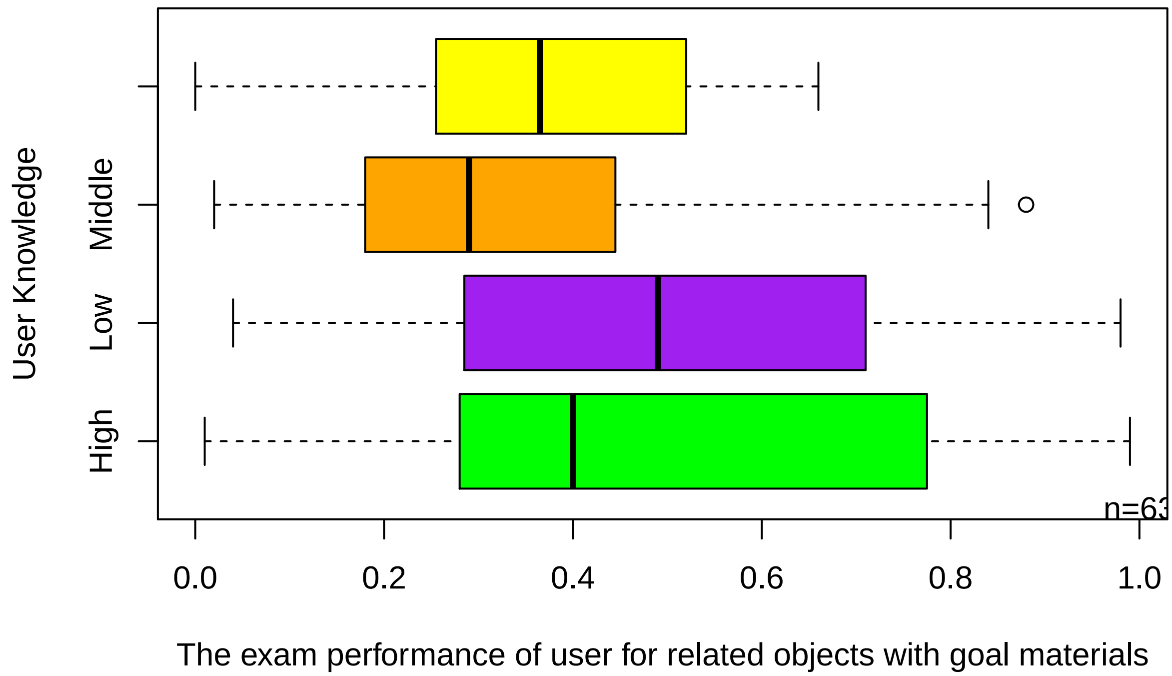

```
# boxplot of STG divided accross UNS values
boxplot2(STG~UNS,data=training_data, horizontal = TRUE,
    xlab="The degree of repetition number of user for goal materials", ylab="User Knowledge", col=c("g
```



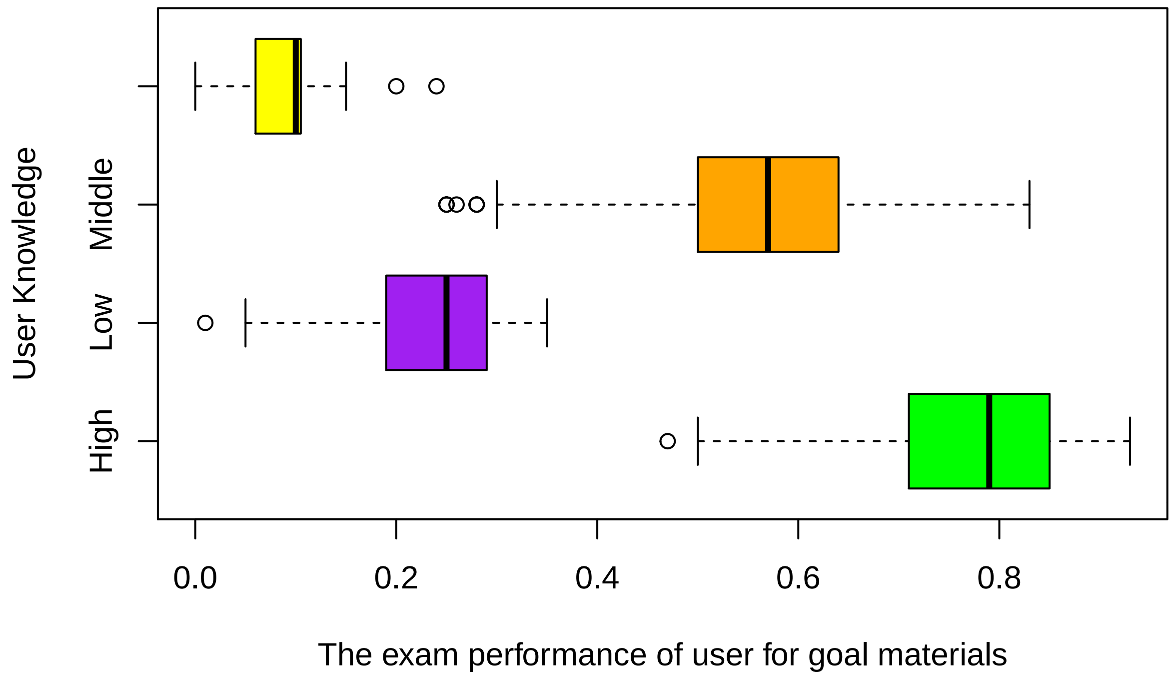

```
# boxplot of STR divided accross UNS values
boxplot2(STR~UNS,data=training_data, horizontal = TRUE,
     xlab="The degree of study time for related objects with goal materials", ylab="User Knowledge", col
```




```
# boxplot of LPR divided accross UNS values
boxplot2(LPR~UNS,data=training_data, horizontal = TRUE,
     xlab="The exam performance of user for related objects with goal materials", ylab="User Knowledge"
```

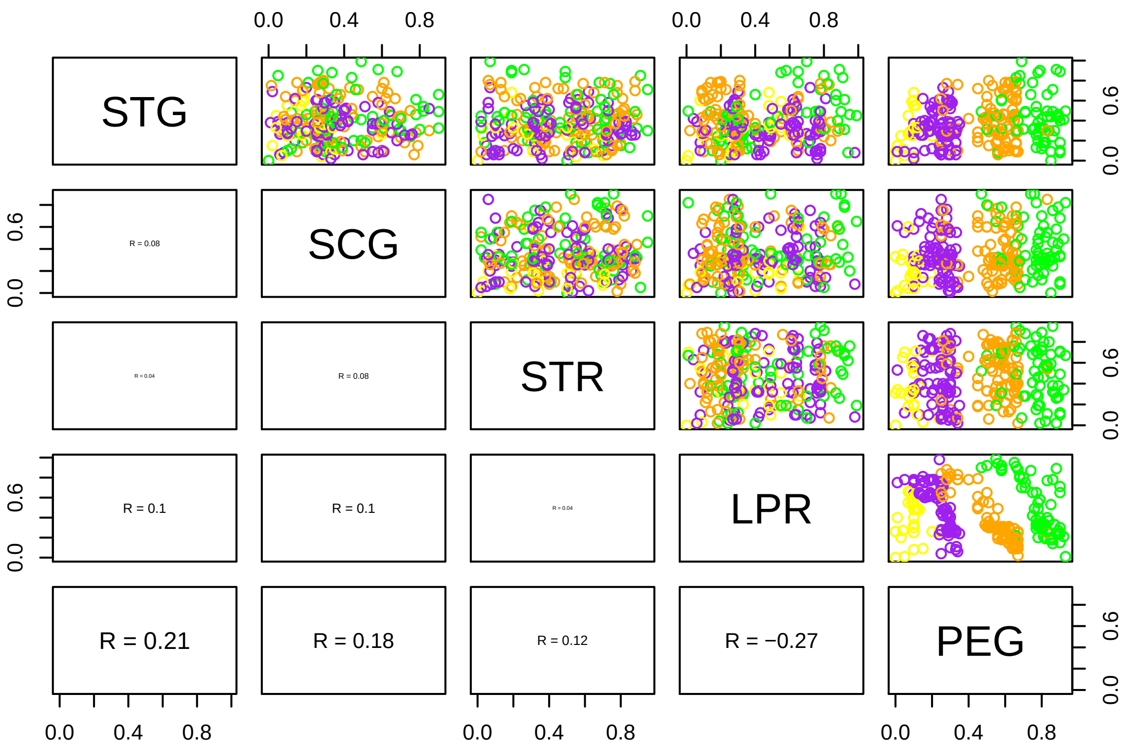



```
# boxplot of PEG divided accross UNS values
boxplot2(PEG~UNS,data=training_data, horizontal = TRUE,
     xlab="The exam performance of user for goal materials", ylab="User Knowledge", col=c("green", "purp
```




```
#Independent variables Scatterplot
my_cols <- c("green", "purple", "orange", "yellow")
#pairs(~STG+SCG+STR+LPR+PEG, data=training_data, col = my_cols[training_data$UNS], upper.panel=NULL)

# Correlation panel
panel.cor <- function(x, y){
    usr <- par("usr"); on.exit(par(usr))
```

```r
    par(usr = c(0, 1, 0, 1))
    r <- round(cor(x, y), digits=2)
    txt <- paste0("R = ", r)
    cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor*r*3)
}
# Customize upper panel
upper.panel<-function(x, y){
  points(x,y, col = my_cols[training_data$UNS])
}
# Create the plots
pairs(~STG+SCG+STR+LPR+PEG, data=training_data, lower.panel = panel.cor, upper.panel = upper.panel)
```




```r
# decision tree
tree1 <- ctree(UNS ~ .,data = training_data)
#plot(tree1) #Review the design
fit1 = predict(tree1, test_data)
table = table(fit1, test_data$UNS)
table
```

```
##
## fit1       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  42      3        5
##   Middle      0   4     30        0
##   very_low    0   0      0       21
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
```

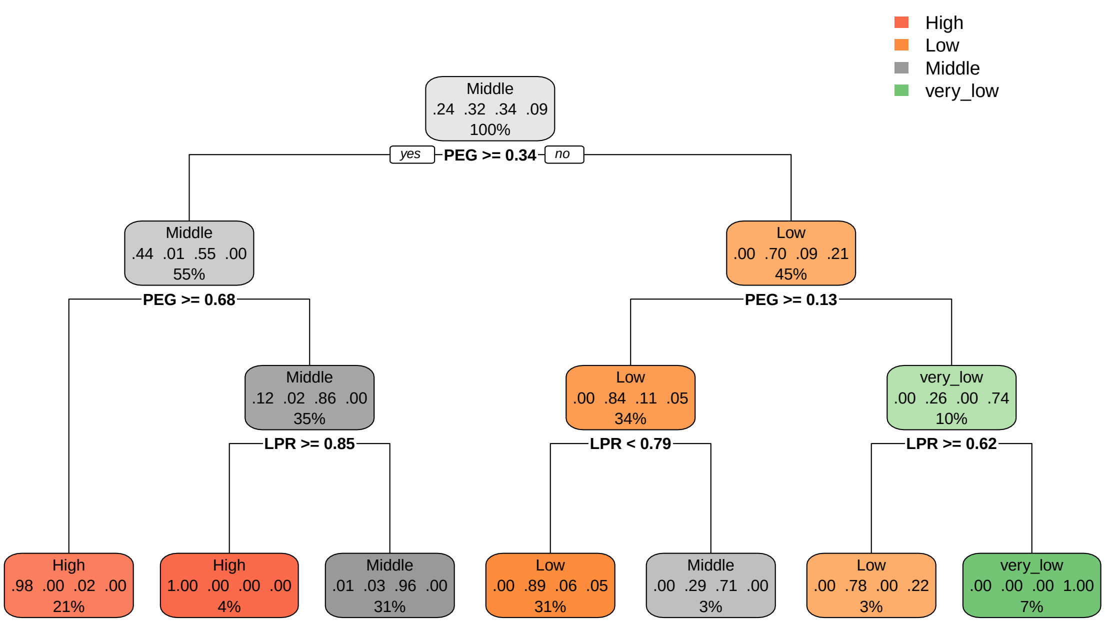
```r
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.9103448
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 0.9750000 0.9873418
## Low      0.9130435 0.8400000 0.8750000
## Middle   0.8823529 0.8823529 0.8823529
## Very Low 0.8076923 1.0000000 0.8936170
```

```r
#recursive partition tree
tree2 <- rpart(UNS ~ ., data = training_data)
rpart.plot(tree2)
```




```r
rpart.rules(tree2)
```

```
##       UNS  High  Low Midd very
##      High [ .98  .00  .02  .00] when PEG >=         0.68
##      High [1.00  .00  .00  .00] when PEG is 0.34 to 0.68 & LPR >= 0.85
##       Low [ .00  .78  .00  .22] when PEG <  0.13         & LPR >= 0.62
##       Low [ .00  .89  .06  .05] when PEG is 0.13 to 0.34 & LPR <  0.79
##    Middle [ .00  .29  .71  .00] when PEG is 0.13 to 0.34 & LPR >= 0.79
```

```
##    Middle [ .01  .03  .96  .00] when PEG is 0.34 to 0.68 & LPR <  0.85
##  very_low [ .00  .00  .00 1.00] when PEG <  0.13         & LPR <  0.62
```

```r
fit2 = predict(tree2, test_data, type = "class")
table = table(fit2, test_data$UNS)
table
```

```
##
## fit2       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  42      3        5
##   Middle      0   4     30        0
##   very_low    0   0      0       21
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.9103448
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 0.9750000 0.9873418
## Low      0.9130435 0.8400000 0.8750000
## Middle   0.8823529 0.8823529 0.8823529
## Very Low 0.8076923 1.0000000 0.8936170
```

```r
# J48 package issues
```

```r
# PART package issues
#hier.part(training_data$UNS, training_data)
```

```r
# Bagging tree NOTE: Interesting we did much better than them here, they did something wrong
tree3 = bagging(UNS ~., data=training_data, coob=TRUE)
fit3 = predict(tree3, test_data)
table = table(fit3, test_data$UNS)

table
```

```
##
## fit3       High Low Middle Very Low
##   High       35   0      1        0
##   Low         0  44      3        3
```

```
##   Middle      4   2     30        0
##   very_low    0   0      0       23
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.9103448
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     0.8974359 0.9722222 0.9333333
## Low      0.9565217 0.8800000 0.9166667
## Middle   0.8823529 0.8333333 0.8571429
## Very Low 0.8846154 1.0000000 0.9387755
```

```r
# Random Forest
tree4 = randomForest(UNS ~., data=training_data)
fit4 = predict(tree4, test_data)
table = table(fit4, test_data$UNS)
table
```

```
##
## fit4       High Low Middle Very Low
##   High       39   0      0        0
##   Low         0  45      3        3
##   Middle      0   1     31        0
##   very_low    0   0      0       23
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)
```

```r
#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.9517241
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 1.0000000 1.0000000
## Low      0.9782609 0.8823529 0.9278351
## Middle   0.9117647 0.9687500 0.9393939
## Very Low 0.8846154 1.0000000 0.9387755
```

```r
# C5.0
tree5 <- C5.0(UNS ~., data=training_data)
fit5 = predict(tree5, test_data)
table = table(fit5, test_data$UNS)
table
```

```
##
## fit5       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  39      3        3
##   Middle      0   5     30        0
##   very_low    0   2      0       23
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.9034483
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 0.9750000 0.9873418
## Low      0.8478261 0.8666667 0.8571429
## Middle   0.8823529 0.8571429 0.8695652
## Very Low 0.8846154 0.9200000 0.9019608
```

```r
table(fit5, test_data$UNS)
```

```
##
## fit5       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  39      3        3
##   Middle      0   5     30        0
##   very_low    0   2      0       23
```

```r
# naive bayes
bayes <- naiveBayes(UNS ~., data=training_data)
fit6 = predict(bayes, test_data)
table = table(fit6, test_data$UNS)

table
```

```
##
## fit6       High Low Middle Very Low
##   High       39   0      0        0
##   Low         0  42      9       10
##   Middle      0   4     25        0
##   very_low    0   0      0       16
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.8413793
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 1.0000000 1.0000000
## Low      0.9130435 0.6885246 0.7850467
## Middle   0.7352941 0.8620690 0.7936508
## Very Low 0.6153846 1.0000000 0.7619048
```

```r
table(fit5, test_data$UNS)
```

```
##
## fit5       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  39      3        3
```

```
##   Middle      0   5     30        0
##   very_low    0   2      0       23
```

```
# Aggregate Data - Add accuracy to each model, compile, add missing algorithms if possible
```

```
# knn
nn4 <- kNN(UNS ~ .,training_data,test_data,norm=FALSE,k=4)
table = table(test_data[,'UNS'],nn4)

table
```

```
##           nn4
##            High Low Middle very_low
##   High       36   0      3        0
##   Low         0  43      2        1
##   Middle      1   6     27        0
##   Very Low    0  13      0       13
```

```
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.8206897
```

```
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     0.9729730 0.9230769 0.9473684
## Low      0.6935484 0.9347826 0.7962963
## Middle   0.8437500 0.7941176 0.8181818
## very_low 0.9285714 0.5000000 0.6500000
```

```
table(fit5, test_data$UNS)
```

```
##
## fit5       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  39      3        3
##   Middle      0   5     30        0
##   very_low    0   2      0       23
```

```
# SVM classification
model <- svm( UNS~., training_data )
res <- predict( model, test_data )
```

```r
table = table(res, test_data$UNS)

table
```

```
##
## res        High Low Middle Very Low
##   High       39   0      0        0
##   Low         0  46      5       10
##   Middle      0   0     29        0
##   very_low    0   0      0       16
```

```r
n = sum(table) # number of instances
nc = nrow(table) # number of classes
diag = diag(table) # number of correctly classified instances per class
rowsums = apply(table, 1, sum) # number of instances per class
colsums = apply(table, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes


precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)

#The accuracy is:
accuracy = sum(diag) / n
accuracy
```

```
## [1] 0.8965517
```

```r
#Here is the performance metrics
data.frame(precision, recall, f1)
```

```
##          precision    recall        f1
## High     1.0000000 1.0000000 1.0000000
## Low      1.0000000 0.7540984 0.8598131
## Middle   0.8529412 1.0000000 0.9206349
## Very Low 0.6153846 1.0000000 0.7619048
```

```r
table(fit5, test_data$UNS)
```

```
##
## fit5       High Low Middle Very Low
##   High       39   0      1        0
##   Low         0  39      3        3
##   Middle      0   5     30        0
##   very_low    0   2      0       23
```