### **Отчёт по работе “Большие данные: кластеризация и классификация”**

Грицаенко Никита, Жуков Александр, Сергеев Георгий, Митрофанова Алина, Чепулис Михаил, Плаксин Даниил

#### **Задача кластеризации**

**Задание**

Имеется набор данных о растениях Армориканской возвышенности (файл ). Требуется провести кластерный анализ данных методом k-медиан с целью их разбиения на групп со сходными признаками (Рассмотреть ). Сделать выводы.

**Данные**

Описание данных: 136 наблюдений, 31 переменная.

#### **Код программы**

library(cluster)  
library(NbClust)  
library(rpart.plot)

## Loading required package: rpart

euc.dist <- function(x1, x2)  
 sqrt(sum((x1 - x2) ^ 2))

**Борьба с NA**

Предложим такой способ борьбы с NA (Not Available) - ячейки матрицы, помеченные как NA, заменим на медианное значение соответствующих ячеек в других прецедентах:

removeNA <- function(matrix) {  
 data\_dim = ncol(matrix)  
 data\_size = nrow(matrix)  
 M = matrix  
 for (i in seq(1, data\_dim)) {  
 med <- median(M[, i], na.rm = TRUE)  
 for (j in seq(1, data\_size)) {  
 if (is.na(M[j, i])) {  
 M[j, i] = med  
 }  
 }  
 }  
 return(M)  
}

**Масштабирование**

normalizeMatrix <- function(matrix) {  
 data\_dim = ncol(matrix)  
 M = matrix  
 for (i in seq(1, data\_dim)) {  
 M[, i] = M[, i] / norm(data.matrix(M[, i]), type = "M")  
 }  
 return(M)  
}

**Кластеризация**

Кластеризуем с помощью функции pam. В качестве критерия кластеризации возвращаем среднеквадратическую ошибку разбиения:

clusterize <- function(matrix, num\_of\_clusters) {  
 data\_dim = ncol(matrix)  
 data\_size = nrow(matrix)  
   
 cl <-  
 pam(  
 matrix,  
 k = num\_of\_clusters,  
 metric = "euclidean",  
 stand = T,  
 keep.diss = TRUE  
 )  
   
 centers = cl$medoids  
 error = 0  
 for (i in seq(1, data\_size)) {  
 min\_dist = 1000000  
   
 for (j in seq(1, num\_of\_clusters)) {  
 dist = euc.dist(matrix[i, ], centers[j, ])  
   
 if (dist < min\_dist) {  
 min\_dist = dist  
 }  
 }  
 error = error + min\_dist ^ 2  
 }  
   
 return(sqrt(error))  
}

**Считывание данных, их обработка**

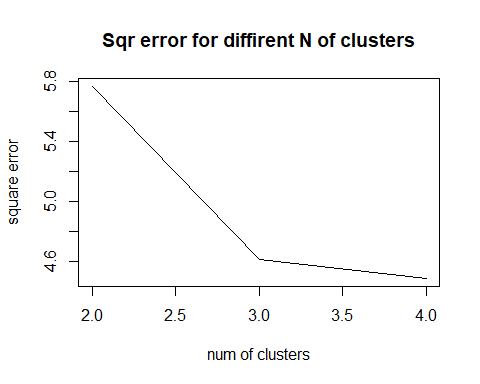
# Read data in matrix M  
path = 'data/plants.dat'  
data.plants <- read.table(  
 path,  
 sep = ';',  
 header = TRUE,  
 na.strings = "NA",  
 stringsAsFactors = T  
)  
  
data.plants$plant.name = NULL  
M = data.matrix(data.plants)  
  
data\_dim = ncol(M)  
data\_size = nrow(M)  
  
# Process data  
M = removeNA(M)  
M = normalizeMatrix(M)

**Выбор компонент для кластеризации**

#Построим матрицу корреляции  
corr\_dm = cor(M, method = "pearson")  
  
#Посмотрим, где корреляция менее всего зависит от других переменных  
corr = 1:data\_dim  
for (i in 1:data\_dim) {  
 corr[i] = sum(abs(corr\_dm[, i]))  
}  
  
variables\_idx = c(5, 7, 30, 1, 6)  
dim = 5  
  
 #Соберём новую матрицу, которая состоит из наименее коррелированных столбцов  
newM <- matrix(0, nrow = data\_size, ncol = dim)  
  
for (i in 1:dim) {  
 newM[, i] = M[, variables\_idx[i]]  
}

**Кластеризация**

# Trying to clustering  
min\_num\_of\_clusters = 2  
max\_num\_of\_clusters = 4  
  
error\_vec = matrix(0, nrow = 1, ncol = 3)  
i = 1  
  
for (k in seq(min\_num\_of\_clusters, max\_num\_of\_clusters)) {  
 error\_vec[i] = clusterize(newM, k)  
 i = i + 1  
}  
  
plot(min\_num\_of\_clusters:max\_num\_of\_clusters,  
 error\_vec,  
 "l",  
 main = "Sqr error for diffirent N of clusters",  
 xlab = "num of clusters",  
 ylab = "square error")



Видим, что по критерию среднеквадратичной ошибки наилучшее число кластеров: 4

**Поиск оптимального числа кластеров с помощью NbClust**

С помощью пакета NbClust можно найти оптимальную схему объединения в кластеры, используя 30 индексов. При этом происходит перебор различных комбинаций числа групп, метрик дистанции и методов кластеризации. Вывод об оптимальном числе классов делается с помощью голосования: берется то число кластеров, за которое “проголосовало” большинство критериев.

# Finding optimal N of clusters  
NbClust(  
 data = newM,  
 diss = NULL,  
 distance = "euclidean",  
 min.nc = min\_num\_of\_clusters,  
 max.nc = max\_num\_of\_clusters,  
 method = "median",  
 index = "all"  
)

\* Among all indices:   
\* 9 proposed 2 as the best number of clusters   
\* 4 proposed 3 as the best number of clusters   
\* 11 proposed 4 as the best number of clusters   
  
 \*\*\*\*\* Conclusion \*\*\*\*\*   
   
\* According to the majority rule, the best number of clusters is 4

***Видим, что наш результат совпал с результатом NbClust.***

### **Задача классификации**

Имеется множество объектов , конечное множество ответов .

Задана выборка и множество известных ответов , вектор – набор признаков, совокупность упорядоченных пар “объект-ответ” – обучающая выборка.

Ставится задача построить решающее правило , которое приближало бы функцию на всем множестве (построить алгоритм, классифицирующий произвольный объект из исходного множества).

#### **Задание**:

Имеется таблица данных о качестве белых вин (Файл ). Требуется методом деревьев по 90% данных построить классификатор и проверить его на 10% приведенных данных. Сделать выводы.

#### **Данные**:

Описание данных: 4898 наблюдений, 12 переменных. Качество вина оценивается переменной , значения которой от 0 (плохое вино) до 10 (самое лучшее вино). Следующая гистограмма отображает исходные данные.

#### **Код программы**

**Библиотеки и функции, необходимые для работы программы** + drawHistogram - создает и отображает гистограмму с заголовком title для таблицы wines. В качестве критерия используется колонка с именем quality. Над каждым столбцом гистограммы отображается соответствующее числовое значение. + makeDataSets - перемешивает данные(строки) таблицы wines случайным образом и создает два набора, содержащие 90% и 10% данных соответственно. Возвращаемое значение - список из двух наборов данных. Случайный закон инициализируется константным значением seed, чтобы результаты воспроизводились от опыта к опыту. + testModel - производит тестирование модели tree на наборах test\_data и train\_data. Предполагается, что обучение модели производится с помощью набора train\_data. + combineClasses - изменяет столбец quality, являющийся фактором таблицы wines. Комбинирует уровни факторов ‘3’, ‘4’, ‘5’, а также ‘7’, ‘8’, ‘9’ в уровни ‘Low’ и ‘High’. Уровень ‘6’ переименовывается в ‘Medium’. **Запустите этот фрагмент кода для корректной работы следующих фрагментов.**

library(repr)  
library(maptree)  
library(tree)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(forcats)  
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

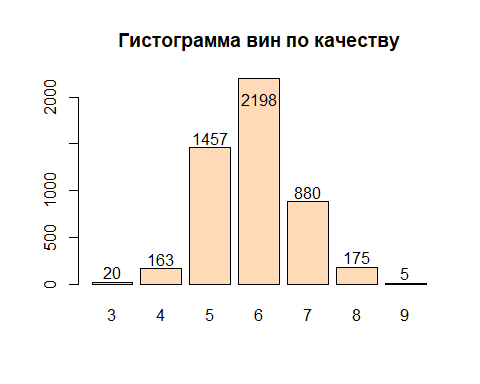
drawHistogram <- function(wines, title){  
 options(repr.plot.width=6, repr.plot.height=6)  
 element\_samples <- summary(wines$quality)  
 midpoints <- barplot(element\_samples, col = "peachpuff1", main = title)  
 abline(h = nrow(wines), lty = 2, col = 2)  
 num\_elements <- dim(data.wine)[1]  
 num\_types <- length(element\_samples)  
 y\_coord <- element\_samples  
 max\_val <- max(element\_samples)  
 for (j in seq(1, num\_types)) {  
 if (element\_samples[j] / max\_val > 0.9){  
 y\_coord[j] <- y\_coord[j] - 0.1 \* max\_val   
 }  
 else{  
 y\_coord[j] <- y\_coord[j] + 0.05 \* max\_val  
 }  
   
 }  
 text(midpoints, y\_coord, labels=element\_samples)  
}  
  
makeDataSets <- function(wines){  
 set.seed(12345) #constant seed for result reproducibility  
 n <- dim(wines)[1]  
 data <- wines[order(runif(n)),]  
 n.train <- as.integer(0.8 \* n)  
 data.train <- data[1:n.train,]  
 data.test <- data[(n.train + 1):n,]  
 return (list(train = data.train, test = data.test))  
}  
  
testModel <- function(tree, test\_data, train\_data){  
   
 predict.test <- predict(tree, test\_data, type = "class")  
 predict.train <- predict(tree, train\_data, type = "class")  
   
 result.test <- table(test\_data$quality, predict.test)  
 result.train <- table(train\_data$quality, predict.train)  
   
 accuracy.test <- sum(diag(result.test)) / sum(result.test)  
 accuracy.train <- sum(diag(result.train)) / sum(result.train)  
 print("test data prediction accuracy: ")  
 print(accuracy.test)  
 print("train data prediction accuracy:")  
 print(accuracy.train)  
   
 #confusion matrix  
 caret::confusionMatrix(test\_data$quality, predict.test)  
   
}  
  
  
combineClasses <- function(wines){  
 df = wines  
 df %>%  
 mutate(quality = fct\_collapse(df$quality,  
 Low = c("3","4","5"),  
 Medium = c("6"),  
 High = c("7","8","9")))->x  
 return(x)  
}

**Считывание данных и предобработка**

# Read data  
data.wine <- read.table('data/winequality-white.csv',   
 sep=';',   
 header=TRUE,   
 na.strings="NA",  
 stringsAsFactors=T)  
  
# Make 'quality' vector to be factor  
data.wine$quality <- as.factor(data.wine$quality)  
  
# Remove 'NA' rows from table.   
# N.B. In file 'data/winequality-white.csv' there is no NA values, so this line is irrelevant  
data.wine <- data.wine[complete.cases(data.wine),]  
  
# Scale all columns in table except 'quality', which is factor column  
data.wine[, -dim(data.wine)[2]] <- scale(data.wine[, -dim(data.wine)[2]])

**Гистограмма распределения вин по качеству**

drawHistogram(data.wine, "Гистограмма вин по качеству")



Видно, что больше всего имеется сведений о вине среднего качества, а о вине низшего и высшего сорта известно мало. Вообще, для задачи классификации такое распределение исходных данных является очень плохим. Классификатор, построенный по этим данным, будет плохо работать.

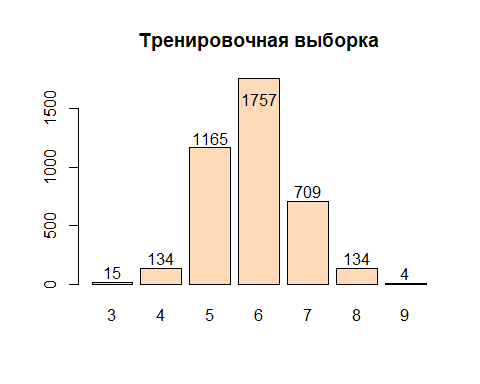
**Разделение данных на тренировочную и тестовую выборки**

Перемешаем данные, разделим их на две группы – тренировочную (90%) и тестовую (10%).

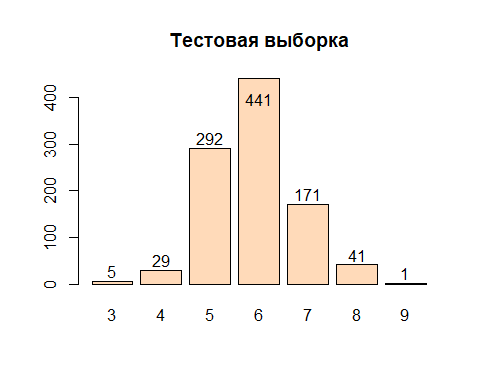
data <- makeDataSets(data.wine)

Получим следующие гистограммы для тренировочной и тестовой выборок.

drawHistogram(data$train, "Тренировочная выборка")



drawHistogram(data$test, "Тестовая выборка")



**Построение модели классификации** Построим дерево решений при помощи tree. Проверим его на выборках, а также построим матрицу сопряженности для построенной модели.

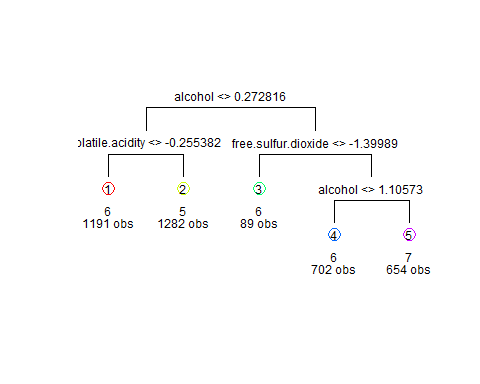
wine.tree <- tree(quality ~., data$train)  
  
testModel(wine.tree, data$test, data$train)

## [1] "test data prediction accuracy: "  
## [1] 0.5193878  
## [1] "train data prediction accuracy:"  
## [1] 0.5096988

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 3 4 5 6 7 8 9  
## 3 0 0 3 2 0 0 0  
## 4 0 0 21 7 1 0 0  
## 5 0 0 185 101 6 0 0  
## 6 0 0 108 259 74 0 0  
## 7 0 0 8 98 65 0 0  
## 8 0 0 2 22 17 0 0  
## 9 0 0 1 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5194   
## 95% CI : (0.4876, 0.5511)  
## No Information Rate : 0.499   
## P-Value [Acc > NIR] : 0.1064   
##   
## Kappa : 0.2568   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8  
## Sensitivity NA NA 0.5640 0.5297 0.39877 NA  
## Specificity 0.994898 0.97041 0.8359 0.6293 0.87026 0.95816  
## Pos Pred Value NA NA 0.6336 0.5873 0.38012 NA  
## Neg Pred Value NA NA 0.7922 0.5733 0.87886 NA  
## Prevalence 0.000000 0.00000 0.3347 0.4990 0.16633 0.00000  
## Detection Rate 0.000000 0.00000 0.1888 0.2643 0.06633 0.00000  
## Detection Prevalence 0.005102 0.02959 0.2980 0.4500 0.17449 0.04184  
## Balanced Accuracy NA NA 0.7000 0.5795 0.63452 NA  
## Class: 9  
## Sensitivity NA  
## Specificity 0.99898  
## Pos Pred Value NA  
## Neg Pred Value NA  
## Prevalence 0.00000  
## Detection Rate 0.00000  
## Detection Prevalence 0.00102  
## Balanced Accuracy NA

Полученное дерево решений представлено на иллюстрации ниже.

options(repr.plot.width = 14, repr.plot.height = 10)  
draw.tree(wine.tree, cex = 0.75)



Построим дерево решений при помощи rpart. Проверим его на выборках, а также построим матрицу сопряженности для построенной модели.

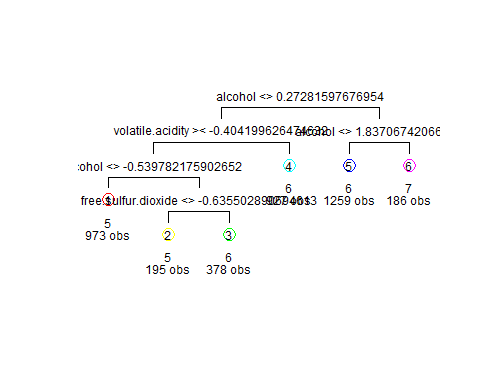
wine.rpart <- rpart(quality ~., data$train)  
testModel(wine.rpart, data$test, data$train)

## [1] "test data prediction accuracy: "  
## [1] 0.555102  
## [1] "train data prediction accuracy:"  
## [1] 0.5370087

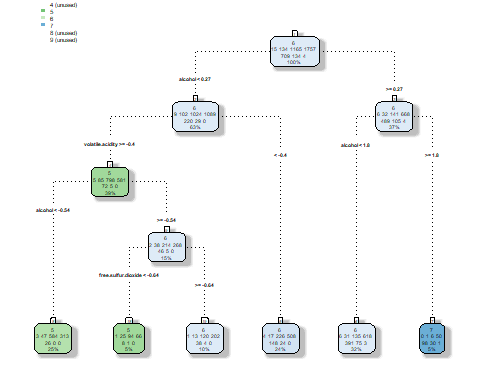
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 3 4 5 6 7 8 9  
## 3 0 0 2 3 0 0 0  
## 4 0 0 19 9 1 0 0  
## 5 0 0 186 105 1 0 0  
## 6 0 0 88 334 19 0 0  
## 7 0 0 5 142 24 0 0  
## 8 0 0 1 38 2 0 0  
## 9 0 0 0 1 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5551   
## 95% CI : (0.5234, 0.5865)  
## No Information Rate : 0.6449   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2706   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8  
## Sensitivity NA NA 0.6179 0.5285 0.51064 NA  
## Specificity 0.994898 0.97041 0.8439 0.6925 0.84244 0.95816  
## Pos Pred Value NA NA 0.6370 0.7574 0.14035 NA  
## Neg Pred Value NA NA 0.8328 0.4471 0.97157 NA  
## Prevalence 0.000000 0.00000 0.3071 0.6449 0.04796 0.00000  
## Detection Rate 0.000000 0.00000 0.1898 0.3408 0.02449 0.00000  
## Detection Prevalence 0.005102 0.02959 0.2980 0.4500 0.17449 0.04184  
## Balanced Accuracy NA NA 0.7309 0.6105 0.67654 NA  
## Class: 9  
## Sensitivity NA  
## Specificity 0.99898  
## Pos Pred Value NA  
## Neg Pred Value NA  
## Prevalence 0.00000  
## Detection Rate 0.00000  
## Detection Prevalence 0.00102  
## Balanced Accuracy NA

Полученное дерево решений представлено на иллюстрации ниже.

draw.tree(wine.rpart, cex = 0.75)



rpart.plot(wine.rpart,   
 type=4,  
 extra=101,   
 box.palette="GnBu",  
 branch.lty=3,   
 shadow.col="gray",   
 nn=TRUE,  
 roundint = FALSE  
)



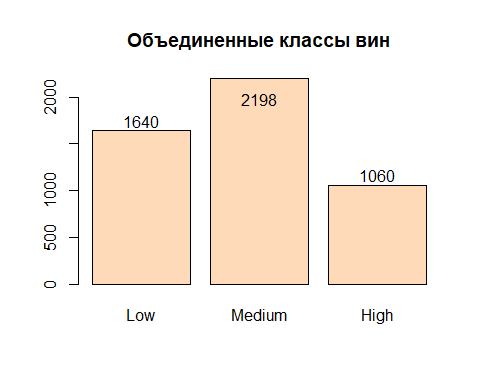
Полученные деревья охватывают не все категории вин из исходных данных, а только 5, 6 и 7. Из-за этого результат применения к тестовой выборке оказывается неудовлетворительным.

Полученный результат следует признать плохим, поскольку дерево решений правильно классифицирует чуть больше 50% данных.

**Улучшение распознавания путем слияния классов качества вин**

Попробуем объединить классы вин 3,4,5 и 7,8,9, чтобы получить более равномерное распределение, чем ранее. Гистограмма данных с новым фактором представлена ниже.

x <- combineClasses(data.wine)  
drawHistogram(x, "Объединенные классы вин")



**Построение модели и тестирование** Построим модель с помощью метода tree. Проверим на тестовой и тренировочной выборке, построим матрицу сопряженности.

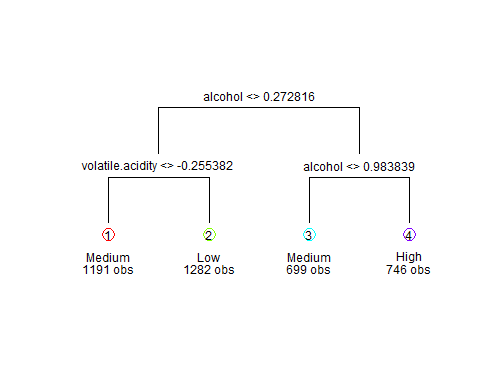
xdata <- makeDataSets(x)  
x.tree <- tree(quality ~., xdata$train)  
  
testModel(x.tree, xdata$test, xdata$train)

## [1] "test data prediction accuracy: "  
## [1] 0.555102  
## [1] "train data prediction accuracy:"  
## [1] 0.5474732

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Low Medium High  
## Low 209 106 11  
## Medium 108 246 87  
## High 11 113 89  
##   
## Overall Statistics  
##   
## Accuracy : 0.5551   
## 95% CI : (0.5234, 0.5865)  
## No Information Rate : 0.4745   
## P-Value [Acc > NIR] : 2.598e-07   
##   
## Kappa : 0.2979   
##   
## Mcnemar's Test P-Value : 0.3341   
##   
## Statistics by Class:  
##   
## Class: Low Class: Medium Class: High  
## Sensitivity 0.6372 0.5290 0.47594  
## Specificity 0.8206 0.6214 0.84363  
## Pos Pred Value 0.6411 0.5578 0.41784  
## Neg Pred Value 0.8180 0.5937 0.87223  
## Prevalence 0.3347 0.4745 0.19082  
## Detection Rate 0.2133 0.2510 0.09082  
## Detection Prevalence 0.3327 0.4500 0.21735  
## Balanced Accuracy 0.7289 0.5752 0.65978

Изображение дерева представлено ниже.

options(repr.plot.width = 14, repr.plot.height = 10)  
draw.tree(x.tree, cex = 0.75)

 Построим модель с помощью метода rpart. Проверим на тестовой и тренировочной выборке, построим матрицу сопряженности.

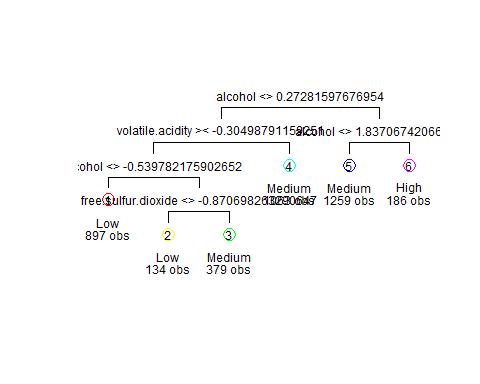
x.rpart <- rpart(quality ~., xdata$train)  
testModel(x.rpart, xdata$test, xdata$train)

## [1] "test data prediction accuracy: "  
## [1] 0.5714286  
## [1] "train data prediction accuracy:"  
## [1] 0.5612557

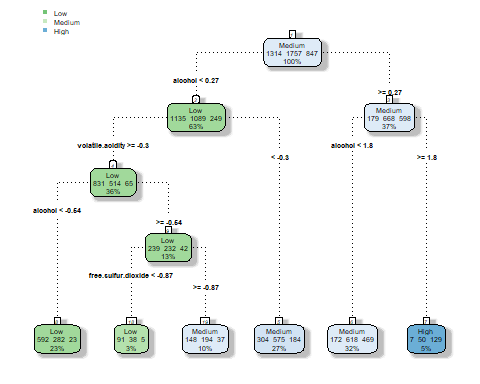
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Low Medium High  
## Low 189 135 2  
## Medium 77 345 19  
## High 6 181 26  
##   
## Overall Statistics  
##   
## Accuracy : 0.5714   
## 95% CI : (0.5398, 0.6027)  
## No Information Rate : 0.6745   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2782   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: Low Class: Medium Class: High  
## Sensitivity 0.6949 0.5219 0.55319  
## Specificity 0.8065 0.6991 0.79957  
## Pos Pred Value 0.5798 0.7823 0.12207  
## Neg Pred Value 0.8731 0.4137 0.97262  
## Prevalence 0.2776 0.6745 0.04796  
## Detection Rate 0.1929 0.3520 0.02653  
## Detection Prevalence 0.3327 0.4500 0.21735  
## Balanced Accuracy 0.7507 0.6105 0.67638

Изображение дерева представлено ниже.

draw.tree(x.rpart, cex = 0.75)



rpart.plot(x.rpart,   
 type=4,  
 extra=101,   
 box.palette="GnBu",  
 branch.lty=3,   
 shadow.col="gray",   
 nn=TRUE,  
 roundint = FALSE  
)

 ####**Заключение**  
По итогам проведенного объединения классов получено незначительное улучшение на 3-4% при распознавании как тестовой, так и тренировочной выборок. Метод rpart даёт результаты чуть лучше, чем метод tree. Кроме того, в библиотеке rpart есть метод rpart.plot для удобного и наглядного изображения дерева решений.