**Наивный Байесовский классификатор**

**Использование пакета ‘e1071’**Функция **naiveBayes** вычисляет условные апостериорные вероятности категориальных переменных при условии независимости признаков с использованием правила Байеса.

**Использование:**

naiveBayes(formula, data, laplace = 0, subset, na.action = na.pass)

naiveBayes(x, y, laplace = 0, ...)

predict(object, newdata, type = c("class", "raw"), ...)

**Аргументы:**

**x** – числовая матрица или фрейм данных

**y** – вектор значений меток классов

**formula** – формула вида **class ~ x1 + x2 +** ....

**data** – фрейм данных

**laplace** – положительный параметр сглаживания Лапласа (0 по умолчанию).

**subset** – индексный вектор, индексирующий обучающую выборку

**na.action** – функция, указывающая, что делать, если пропущенные данные имеют место

**object** – объект класса "naiveBayes"

**newdata** – фрейм данных новых объектов для тестирования

**type** – если "raw", то возвращаются условные апостериорные вероятности для каждого класса, иначе класс с максимальной вероятностью.

**Выходные значения:**

**naiveBayes** возвращает объект-список, включающий элементы:

**apriori** – распределение классов

**tables** – список таблиц по одной для каждой переменной, которые содержат условные вероятности переменных при условии класса.

**Пример 1.**

**Обучающее множество: Tic Tac Toe (крестики-нолики)**

"x" начинает первым. Цель: "победа x"

**Признаки (категориальные – 3 значения)**:

1. top-left-square: {x,o,b}   
2. top-middle-square: {x,o,b}   
3. top-right-square: {x,o,b}   
4. middle-left-square: {x,o,b}   
5. middle-middle-square: {x,o,b}   
6. middle-right-square: {x,o,b}   
7. bottom-left-square: {x,o,b}   
8. bottom-middle-square: {x,o,b}   
9. bottom-right-square: {x,o,b}

**3 класса**: {positive,negative}

#install.packages("e1071")

library(e1071)

### Naive Bayesian (данные категориальные) #########################

# 1 #############################################################

# импортируем данные в R

# установить параметр stringsAsFactors = TRUE,

# так как все данные - категориальные

A\_raw <- read.table("Tic\_tac\_toe.txt", sep = ",", stringsAsFactors = TRUE)

# число строк в базе

n <- dim(A\_raw)[1]

# Создан фрейм, который можно просмотреть, используя str(A\_raw).

# Имеется 9 столбцов признаков V1-V9 и V10 (класс) и

# все имеют один и тот же тип Factor.

# 2 #############################################################

# Создание обучающей и тестирующей выборки

# Скажем, имеем n примеров в исходной выборке,

# используем 80% для обучения и оставшиеся - для тестирования.

# Устанавливаем базу генерации случайных чисел и рандомизируем выборку

set.seed(12345)

A\_rand <- A\_raw[ order(runif(n)), ]

# разделим данные на обучающие и тестирующие

nt <- as.integer(n\*0.8)

A\_train <- A\_rand[1:nt, ]

A\_test <- A\_rand[(nt+1):n, ]

# Можно убедиться, какой имеется процент каждого

# класса V2 в обучающей и тестирующей выборке

prop.table(table(A\_train$V10))

prop.table(table(A\_test$V10))

# 3 ############################################################

# Используем Наивный Байесовский классификатор из пакета e1071

# A\_classifier <- naiveBayes(A\_train[,-10], A\_train$V10)

# Другой вариант классификатора

A\_classifier <- naiveBayes(V10 ~ ., data = A\_train)

# 4 ############################################################

# Теперь оценим полученную модель:

A\_predicted <- predict(A\_classifier, A\_test)

# Используем table для сравнения прогнозируемых значений с тем, что есть

table(A\_predicted, A\_test$V10)

**Пример 2.**

**Обучающее множество: Данные о спаме e-mail сообщений**

Данные собраны Hewlett-Packard Labs, которая классифицировала 4601 e-mail сообщений как спам или не спам. 57 признаков, содержащих частоты определенных слов, соответствующих названию признака и букв в сообщениях. Данные содержат 2788 e-mail сообщений, классифицируемых как "не спам" (0) and 1813 сообщений, классифицируемых как "спам" (1). Часть признаков – частоты определенных слов, соответствующих названию признака. Часть признаков связана с числом заглавных букв в письме.

library(kernlab)

library(e1071)

data(spam)

## Посмотрим, какие признаки используются и их значения

spam[0:1,]

## Случайным образом выбираем 20 сообщений для тестирования,

## точнее индексы 20-ти тестов

idx <- sample(1:dim(spam)[1], 20)

spamtrain <- spam[-idx, ]

spamtest <- spam[idx, ]

## Обучаем классификатор

model <- naiveBayes(type ~ ., data = spamtrain)

predict(model, spamtest)

table(predict(model, spamtest), spamtest$type)

predict(model, spamtest, type = "raw")

**Пример 3.**

**Обучающее множество: Iris**

**Признаки:**

1. длина чашелистика (sepal) в см

2. ширина чашелистика (sepal) в см

3. длина лепестка в см

4. ширина лепестка в см

**Классы:**

-- Iris Setosa

-- Iris Versicolour

-- Iris Virginica

#install.packages("e1071")

library(e1071)

data(iris) # load iris dataset

pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)",

pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])

head(iris,n=12)

summary(iris)

classifier<-naiveBayes(iris[,1:4], iris[,5])

predicted.classes <- predict(classifier, iris[,-5])

head(predicted.classes,n=12)

table(predicted.classes, iris[,5], dnn=list('predicted','actual'))

classifier$apriori / sum(classifier$apriori) # the prior distribution for the classes

classifier$tables$Petal.Length

plot(0:3, xlim=c(0.5,7), col="red", ylab="density",type="n", xlab="Petal Length",main="Petal length distribution for each species")

curve(dnorm(x, classifier$tables$Petal.Length[1,1], classifier$tables$Petal.Length[1,2]), add=TRUE, col="red")

curve(dnorm(x, classifier$tables$Petal.Length[2,1], classifier$tables$Petal.Length[2,2]), add=TRUE, col="blue")

curve(dnorm(x, classifier$tables$Petal.Length[3,1], classifier$tables$Petal.Length[3,2]), add=TRUE, col ="green")

legend("topright", c("setosa", "versicolor", "virginica"), col = c("red","blue","green"), lwd=1)

observation <- data.frame(Sepal.Length = 5.0,

Sepal.Width = 3.2,

Petal.Length = 1.5,

Petal.Width = 0.3) # this observation lies within Setosa cluster

# type="raw" shows the probabilities

predict(classifier, observation, type="raw")

**Задание**

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации или на вероятность ошибочной классификации в примере крестики-нолики и примере о спаме e-mail сообщений.

2. Сгенерируйте 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что первые 50 точек (class -1) имеют параметры: мат. ожидание X1 равно 10, мат. ожидание X2 равно 14, среднеквадратические отклонения для обеих переменных равны 4. Вторые 50 точек (class +1) имеют параметры: мат. ожидание X1 равно 20, мат. ожидание X2 равно 18, среднеквадратические отклонения для обеих переменных равны 3. Построить соответствующие диаграммы, иллюстрирующие данные. Построить байесовский классификатор и оценить качество классификации.

3. Разработать байесовский классификатор для данных **Титаник (Titanic dataset) -** <https://www.kaggle.com/c/titanic>

Исходные обучающие данные для классификации – в файле Titanic\_train.csv

Данные для тестирования – в файле Titanic\_test.csv

Использовать функцию read.csv для чтения данных из csv-файлов.

**Классы:**

survival Выжил (0 = No; 1 = Yes)

**Признаки:**

pclass Класс каюты (1 = 1st; 2 = 2nd; 3 = 3rd)

name Имя

sex Пол

age Возраст

sibsp Число братьев-сестер/муж-жена на борту

parch Число родителей/детей на борту

ticket Номер билета

fare Стоимость билета

cabin Каюта

embarked Порт посадки (C = Cherbourg; Q = Queenstown; S = Southampton)

**Специальные отметки**:

Pclass: 1st ~ Верхний; 2nd ~ Средний; 3rd ~ Нижний

Age – в годах; дробный, если возраст меньше одного года