Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

И.И. Кручинин
(к.т.н. доцент)

ЛАБОРАТОРНАЯ РАБОТА № 4
по курсу «Методы машинного обучения»

Логические методы классификации многомерных объектов пересекающихся классов

Калуга

2018

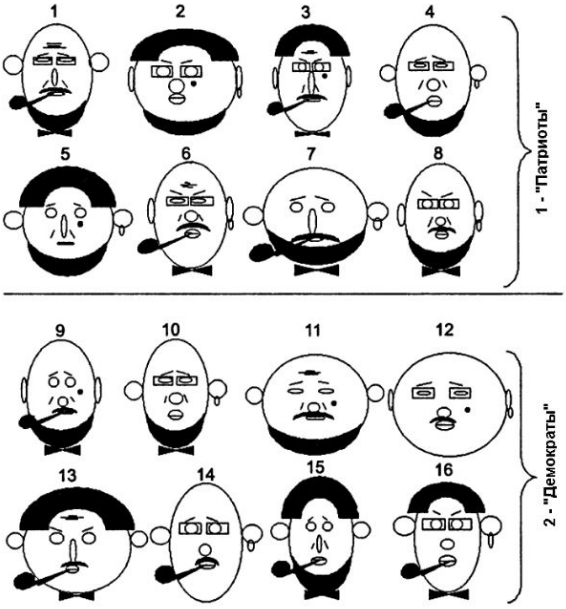
Теоретические основы.

Логические методы классификации

Классификация на основе логических высказываний основывается, как правило, на собранных закономерных явлениях, основанных на большом количестве непротиворечивых фактах. Всякая закономерность классифицирует лишь некоторую часть объектов. Объединив определённое количество закономерностей в композицию, можно получить алгоритм, способный классифицировать любые объекты. Логическими алгоритмами классификации будем называть композиции легко интерпретируемых закономерностей

Методы поиска логических закономерностей в бинарных матрицах наблюдений апеллируют к информации, заключенной в различных сочетаниях их значений. Для признаков, измеренных в альтернативных шкалах, их значения xi=0 или xi=1 рассматриваются как элементарные события, что позволяет сформулировать решающие правила на простом и понятном человеку языке логических высказываний, таких как ЕСЛИ {(событие 1) и (событие 2) и ... и (событие N)} ТО ....

Предположим, что некая избирательная комиссия озаботилась выделением характерных черт групп электората, голосующих за различные общественные платформы. На представленном рисунке схематично изображены лица людей, относящихся к двум классам (для определенности пусть это будут “Патриоты” и “Демократы”). Ставится задача найти комбинации признаков, характеризующие это разделение.



Прежде всего, выделим признаки, характеризующие изображенные лица. Это следующие характеристики:

x1 (голова) – круглая (1) или овальная (0);
x2 (уши) – оттопыренные (1) или прижатые (0);
x3 (нос) – круглый (1) или длинный (0);
x4 (глаза) – круглые (1) или узкие (0);
x5 (лоб) – с морщинами (1) или без морщин (0);
x6 (носогубная складка) – есть (1) или нет (0);
x7 (губы) – толстые (1) или тонкие (0);
x8 (волосы) – есть (1) или нет (0);
x9 (усы) – есть (1) или нет (0);
x10 (борода) – есть (1) или нет (0);
x11 (очки) – есть (1) или нет (0);
x12 (родинка на щеке) – есть (1) или нет (0);
x13 (бабочка) – есть (1) или нет (0);
x14 (брови) – подняты кверху (1) или опущены книзу (0);
x15 (серьга) – есть (1) или нет (0);
x16 (курительная трубка) – есть (1) или нет (0).

Загрузим исходную матрицу данных, соответствующую изображенным лицам, строки которой соответствуют объектам (n = 16), а столбцы - выделенным бинарным признакам (m = 16). Объекты с номерами 1-8 относятся к классу 1, а с номерами 9-16 - к классу 2.

```
DFace <- read.delim(file = "data/Faces.txt", header = TRUE, row.-
names = 1)

head(DFace, 8)
```

Классификация лиц может быть произведена с помощью логических высказываний 1 и 2:

1. ЕСЛИ {(голова овальная) и (есть носогубная складка) и (есть очки) и (есть трубка)} ИЛИ {(глаза круглые) и (лоб без морщин) и (есть борода) и (есть серьга)} ТО ("Патриот")
2. ЕСЛИ {(нос круглый) и (лысый) и (есть усы) и (брови подняты кверху)} ИЛИ {(оттопыренные уши) и (толстые губы) и (нет родинки на щеке) и (есть бабочка)} ТО ("Демократ").

Математическая запись этих правил выглядит следующим образом:
[(x1=0)∧(x6=1)∧(x11=1)∧(x16=1)]∨[(x4=1)∧(x5=0)∧(x10=1)∧(x15=1)]⇒ω1[
(x1=0)∧(x6=1)∧(x11=1)∧(x16=1)]∨[(x4=1)∧(x5=0)∧(x10=1)∧(x15=1)]⇒ω1
[(x3=1)∧(x8=0)∧(x9=1)∧(x14=1)]∨[(x2=1)∧(x7=1)∧(x12=0)∧(x13=1)]⇒ω2.
[(x3=1)∧(x8=0)∧(x9=1)∧(x14=1)]∨[(x2=1)∧(x7=1)∧(x12=0)∧(x13=1)]⇒ω2.
Здесь значки ∨∨ - конъюнкция (“и”), ∧∧ - дизъюнкция (“или”), ⇒⇒ - импликация (“если, то”).

Алгоритм “Кора”

Будем использовать логический метод распознавания, известный под названием **алгоритм “Кора”,** который в свое время широко использовался в геологоразведочных работах и скрининге лекарственных препаратов (Бонгард, 1967; Вайнцвайг, 1973). В детерминистической версии алгоритма обучающая выборка x , предварительно разделенная на два класса (1 и 2), многократно просматривается и из всего множества высказываний выделяются так называемые непротиворечивые логические высказывания $\Psi(x,\tau)$, покрывающие все множество примеров. Непротиворечивым высказыванием для каждого класса считается конъюнкция, которая встречается с вероятностью не менее τ только в одном классе и ни разу не встречается в другом.

Представленный ниже скрипт выполняет перебор всех возможных комбинаций столбцов xi , i=1,…,m=16 по 2, 3 и maxKSize = 4 с использова-

нием функции combn(). Для каждой комбинации столбцов перебираются все возможные варианты событий (x_i = 0 или x_i = 1). Для сокращения объема вычислений подмножества исходной матрицы трансформировались в векторы символьных бинарных переменных, а комбинации значений xi выражались наборами “битовых масок” (например, "000", "001", "010", "011", "100", "101", "110", "111").
В ходе перебора конъюнкции, отобранные по совокупности условий, будем сохранять в трех глобальных объектах класса list, для чего используем оператор глобального присваивания <<-. Определим предварительно несколько функций:

```
# Преобразование числа в набор битов (5 -> "0101")
number2binchar <- function(number, nBits) {
   paste(tail(rev(as.numeric(intToBits(number))), nBits),
         collapse = "")
}

#  Поиск конъюнкций по набору битовых масок
MaskCompare <- function(Nclass, KSize, BitMask,
                        vec_pos, vec_neg, ColCom) {
   nK <- sapply(BitMask, function(x) {
      if (sum(x == vec_neg) > 0) return (0)
      if (minNum > (countK = sum(x == vec_pos))) return(0)
      #  Сохранение конъюнкции в трех объектах list
      Value.list[[length(Value.list) + 1]] <<-
         list(Nclass = Nclass, KSize = KSize,
              countK = countK, Bits = x)
      ColCom.list[[length(ColCom.list) + 1]] <<- list(ColCom)
      RowList.list[[length(RowList.list) + 1]] <<-
         list(which(vec_pos %in% x))
      return(countK) } )
}
```

Зададим минимальную частоту встречаемости конъюнкций minNum = 4 (т.е. $\tau = 50\%$) и выполним формирование всех логических правил для рассматриваемого примера:

```
DFace <- read.delim(file = "data/Faces.txt",
                    header = TRUE, row.names = 1)
maxKSize <- 4
minNum <- 4

#  Списки для хранения результатов
Value.list <- list()   # Nclass, KSize, BitMask, countK
```

```
ColCom.list <- list()  # Наименования переменных ColCom
RowList.list <- list()  # Номера индексов строк RowList

#  Перебор конъюнкций разной длины
for (KSize in 2:maxKSize) {
   BitMask <- sapply(0:(2^KSize - 1),
                  function(x) number2binchar(x, KSize))
   cols <-  combn(colnames(DFace[, -17]), KSize)

   for (i in 1:ncol(cols))  {
      SubArr <- DFace[, (names(DFace) %in% cols[, i])]
      vec1 <- apply(SubArr[DFace$Class == 1, ],1,
                  function(x) paste(x, collapse = ""))
      vec2 <- apply(SubArr[DFace$Class == 2,], 1,
                  function(x) paste(x, collapse = ""))
      MaskCompare(1, KSize, BitMask, vec1, vec2, cols[, i])
      MaskCompare(2, KSize, BitMask, vec2, vec1, cols[, i])
   }
}
#  Создание результирующей таблицы
DFval = do.call(rbind.data.frame, Value.list)
nrow = length(Value.list)
DFvar <- as.data.frame(matrix(NA, ncol = maxKSize + 1, nrow = nrow,
                        dimnames = list(1:nrow, c(
                           paste("L", 1:maxKSize, sep = ""),
                           "Объекты:"))))
for (i in 1:nrow) {
   Varl <- unlist(ColCom.list[[i]])
   DFvar[i, 1:length( Varl)] <- Varl
   Objl <- unlist(RowList.list[[i]])
   DFvar[i, maxKSize + 1] <- paste(Objl, collapse = " ")
}

DFvar[is.na(DFvar)] <- " "
DFout <- cbind(DFval, DFvar)

#  Вывод результатов
print("Конъюнкции класса 1")
## [1] "Конъюнкции класса 1"
DFout[DFout$Nclass == 1, ]
print("Конъюнкции класса 2")
```

[1] "Конъюнкции класса 2"
DFout[DFout$Nclass == 2,]
Результат, содержащий логические высказывания для каждого класса (Nclass) выглядит стандартным образом, где KSize - длина коньюнкции, Bits - ее битовая маска, L1-L4 - наименования исходных переменных, countK - встречаемость конъюнкции на объектах своего класса.

Существует возможность использовать сгенерированные конъюнкции для экзамена тестируемых примеров по принципу голосования. Чтобы классифицировать новое наблюдение x , подсчитывается число отобранных конъюнкций Lk , характерных для каждого k -го класса, которые верны для тестируемого бинарного вектора. Если Lk является максимальным из всех, то принимается решение о принадлежности объекта k -му классу.
Алгоритм “Кора” является достаточно трудоемким, поскольку при отборе конъюнкций необходим полный или частично направленный перебор. Здесь может быть полезно использовать генетический алгоритм (genetic algorithm).

Генетический алгоритм
Генетический алгоритм (ГА) представляет собой структуру иерархического поиска. ГА создает набор возможных модельных решений и как в процессе эволюции Вселенной, отсеивает неудачные решения. ГА генерирует популяции, состоящие из индивидов – хромосом. Хромосомы ранжируются в порядке значимости, оцениваются и помечаются конкретными значениями. Лучшие хромосомы пытаются создать новые прогрессивные популяции, неудачные индивиды при этом отсеиваются.

Например, представленная таблица иллюстрирует характеристики пунктов которыми можно оперировать при классификации.

item Ниименование	**Survivalpoints Набранные баллы**	**Weight Вес**
pocketknife	10.00	1.00
beans	20.00	5.00
potatoes	15.00	10.00
unions	2.00	1.00
sleeping bag	30.00	7.00
rope	10.00	5.00

compass	30.00	1.00

В языке R используется пакет genalg для реализации генетического алгоритма. Затем необходимо установить пакет ggplot2 для визуализации полученных результатов.

```
library(genalg)
library(ggplot2)

dataset <- data.frame(item = c("pocketknife", "beans", "potatoes", "unions",
   "sleeping bag", "rope", "compass"), survivalpoints = c(10, 20, 15, 2, 30,
   10, 30), weight = c(1, 5, 10, 1, 7, 5, 1))
weightlimit <- 20
```

Для примера используем генную бинарную комбинацию: **1001100**.
Каждое двоичное значение в текстовом виде представляет принадлежность индивида к определенному классу. Значение 1 позволяет ссылаться на возможность перевода специфического пункта в элитный набор (knapsack), если же значение 0, то отмеченный пункт остается в прежнем наборе. В качестве примера конфигурации генов представим набор следующих генов:

```
chromosome = c(1, 0, 0, 1, 1, 0, 0)
dataset[chromosome == 1, ]
##          item survivalpoints weight
## 1  pocketknife            10      1
## 4      unions             2      1
## 5 sleeping bag            30      7
```

Мы можем проверить суммарное количество пунктов surivival points:

```
cat(chromosome %*% dataset$survivalpoints)
## 42
```

Выше мы уже присвоили значение для набора хромосом генной конфигурации. Для этого была определена оценочная функция evaluation function. Генетический алгоритм *genalg* пытается оптимизировать краевые значения (минимумы или максимумы). Поэтому значение переменной рассчитывается увеличением и умножением на -1. Набор – конфигурация, которая может вызвать увеличение the weight constraint returns a value of 0 (a higher value can also be given).
Определим оценочную evaluation функцию:

```
evalFunc <- function(x) {
   current_solution_survivalpoints <- x %*% dataset$survivalpoints
   current_solution_weight <- x %*% dataset$weight

   if (current_solution_weight > weightlimit)
      return(0) else return(-current_solution_survivalpoints)
}
```

Далее, мы будем менять число повторений – итераций, перепроектируем и запустим на выполнение ГА модель.

```
iter = 100
GAmodel <- rbga.bin(size = 7, popSize = 200, iters = iter, mutationChance =
0.01,
   elitism = T, evalFunc = evalFunc)
cat(summary.rbga(GAmodel))

## GA Settings – Настройки модели ГА
##   Type              = binary chromosome – бинарные двоичные хромосомы
##   Population size       = 200 – размер популяции
##   Number of Generations = 100 – число поколений (генераций)
##   Elitism               = TRUE (критерий  оценки)
##   Mutation Chance       = 0.01 (параметр мутации)
##
## Search Domain
##   Var 1 = [,]
##   Var 0 = [,]
##
## GA Results - Результаты
##   Best Solution : 1 1 0 1 1 1 1
```

Наилучшим решением является вариант **1111101**. Используем данный набор для дальнейших исследований:

```
solution = c(1, 1, 1, 1, 1, 0, 1)
dataset[solution == 1, ]
##          item survivalpoints weight
## 1  pocketknife            10      1
## 2       beans            20      5
## 3     potatoes            15     10
## 4      unions              2      1
## 5 sleeping bag            30      7
```

```
## 7     compass          30     1
```

Получим полный список пунктов survival points.

```
# solution vs available
cat(paste(solution %*% dataset$survivalpoints, "/", sum(dataset$survival-
points)))
## 107 / 117
```

Визуализируем работу модели:.

```
animate_plot <- function(x) {
   for (i in seq(1, iter)) {
          temp <- data.frame(Generation = c(seq(1, i), seq(1, i)), Variable =
c(rep("mean",
                i), rep("best", i)), Survivalpoints = c(-GAmodel$mean[1:i], -
GAmodel$best[1:i]))

       pl <- ggplot(temp, aes(x = Generation, y = Survivalpoints, group = Vari-
able,
         colour = Variable)) + geom_line() + scale_x_continuous(limits = c(0,
             iter)) + scale_y_continuous(limits = c(0, 110)) + geom_hline(y =
max(temp$Survivalpoints),
         lty = 2) + annotate("text", x = 1, y = max(temp$Survivalpoints) +
         2, hjust = 0, size = 3, color = "black", label = paste("Best solution:",
         max(temp$Survivalpoints))) + scale_colour_brewer(palette = "Set1") +
         opts(title = "Evolution Knapsack optimization model")

      print(pl)
   }
}

# in order to save the animation  - сохраним результаты анимации
library(animation)
saveMovie(animate_plot(), interval = 0.1, outdir = getwd())
```

Алгоритм CART

Деревья решений (Breiman at al., 1984; Quinlan, 1986) осуществляют разбиение пространства объектов в соответствии с некоторым набором правил разбиения (splitting rule). Эти правила являются логическими утверждениями в отношении той или иной переменной и могут быть истинными или ложными. Ключевыми здесь являются три обстоятельства: а) правила позволяют реализовать последовательную дихотомическую сегментацию данных, б) два объекта считаются похожими, если они оказываются в одном и том же сегменте разбиения, в) на каждом шаге разбиения увеличивается количество информации относительно исследуемой переменной (отклика).

Алгоритм CART (Classification and Regression Tree) рекурсивно делит исходный набор данных на подмножества, которые становятся все более и более гомогенными относительно определенных признаков, в результате чего формируется древовидная иерархическая структура. Деление осуществляется на основе традиционных логических правил в виде ЕСЛИ (А) ТО (В), где А - некоторое логическое условие, а В - процедура деления подмножества на две части, для одной из которых условие А истинно, а для другой - ложно. Примеры условий: Xi==F, Xi <= V; Xi >= V и др., где Xi – один из предикторов исходной таблицы, F - выбранное значение категориальной переменной, V - специально подобранное опорное значение (порог).

На первой итерации корневой узел дерева связывается с наиболее оптимальным условным суждением, и все множество объектов делится на две группы. От каждого последующего узла-родителя к узлам-потомкам также может отходить по две ветви, в свою очередь связанные с граничными значениями других наиболее подходящих переменных и определяющие правила дальнейшего разделения (splitting criterion). Конечными узлами дерева являются “листья”, соответствующие найденным решениям и объединяющие все разделенные на группы объекты обучающей выборки. Общее правило выбора опорного значения для каждого узла построенного дерева можно сформулировать следующим образом: “выбранный признак должен разбить множество $X*$ так, чтобы получаемые в итоге подмножества $X*k,k=1,2,\ldots,p$, состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому”.

В качестве примера рассмотрим построение дерева CART, прогнозирующего обилие водорослей группы a1 в зависимости от гидрохимических показателей воды и условий отбора проб в различных водотоках.

Используем сначала пакет rpart, для работы с которым обычно применяется двухшаговая процедура: функция rpart() устанавливает связи между зависимой и независимыми переменными и формирует бинарное дерево, а функция prun() выполняет обрезание лишних ветвей.

```
rpart(formula, data, weights, subset, na.action = na.rpart, method,
    model = FALSE, x = FALSE, y = TRUE, parms, control, cost, …)
                    method =  "anova", "poisson", "class" или "exp"
```

```
load(file = "algae.RData") # Загрузка таблицы algae - раздел 4.1
library(rpart)
(rt.a1 <- rpart(a1 ~ ., data = algae[, 1:12]))
## n= 200
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 200 90694.880 16.923500
##    2) PO4>=43.818 148 31359.210  8.918919
##      4) Cl>=7.8065 141 21678.580  7.439716
##        8) oPO4>=51.118 85  3455.770  3.801176 *
##        9) oPO4< 51.118 56 15389.430 12.962500
##         18) mnO2>=10.05 24  1248.673  6.716667 *
##         19) mnO2< 10.05 32 12502.320 17.646870
##           38) NO3>=3.1875 9   257.080  7.866667 *
##           39) NO3< 3.1875 23 11047.500 21.473910
##             78) mnO2< 8 13  2919.549 13.807690 *
##             79) mnO2>=8 10  6370.704 31.440000 *
##      5) Cl< 7.8065 7  3157.769 38.714290 *
##    3) PO4< 43.818 52 22863.170 39.705770
##      6) mxPH< 7.87 28 11636.710 32.875000
##       12) oPO4>=3.1665 14  1408.304 23.978570 *
##       13) oPO4< 3.1665 14  8012.309 41.771430 *
##      7) mxPH>=7.87 24  8395.785 47.675000
##       14) PO4>=15.177 12  3047.517 38.183330 *
##       15) PO4< 15.177 12  3186.067 57.166670 *
```

Приведенной командой мы построили полное дерево без обрезания ветвей, состоящее из 9 узлов и 10 листьев, обозначенных в приведенном протоколе разбиения символом *. В каждой строке представлены по по-

рядку: условие разделения, число наблюдений, соответствующих этому условию, девианс (в данном случае - это эквивалент суммы квадратов отклонений от группового среднего) и среднее значение отклика для выделенной ветви. Например, перед первой итерацией общее множество из 200 наблюдений имеет среднее значение m = 16.92 при девиансе D = 90694. При PO4>=43.8 это множество делится на две части: 2) 148 наблюдений (m =8.92, D = 31359) и 3) 52 наблюдения с высоким уровнем обилия водорослей (m =39.7, D = 22863). Дальнейшие разбиения каждой из этих двух частей аналогичны.

Разумеется, лучший вариант – представить дерево графически. Популярны три варианта визуализации с использованием различных функций: plot(), prettyTree() из пакета DMwR и prp() из чрезвычайно продвинутого пакета rpart.plot :

```
prettyTree(rt.a1)
```

Полезно также проследить изменение перечисленных выше статистических критериев по мере выращивания дерева:

```
printcp(rt.a1)
```

Функция rpart() и другие функции из пакета rpart имеют собственные возможности выполнить перекрестную проверку и оценить ее ошибку при различных значениях штрафа за сложность модели:

```
set.seed(505) # для воспроизводимости примера
#   Снижаем порог штрафа за сложность с шагом .005
rtp.a1 <- rpart(a1 ~ ., data = algae[, 1:12],
              control = rpart.control(cp = .005))
#  График зависимости относительных ошибок от числа узлов
plotcp(rtp.a1)
with(rtp.a1, {lines(cptable[, 2] + 1, cptable[, 3], type = "b", col = "red")
   legend("topright", c("Ошибка обучения",
                        "Ошибка крос-проверки (CV)", "min(CV ошибка)+SE"),
          lty = c(1, 1, 2), col = c("red", "black", "black"), bty = "n") })
```

Видно, что минимум относительной ошибки при перекрестной проверке приходится на значение cp = 0.024.6 Выполним обрезку дерева при этом значении:

```
rtp.a1 <- prune(rtp.a1, cp = 0.024)
prettyTree(rtp.a1)
```

Бэггинг

Применим методы бэггинга и случайного леса к прогнозированию данных по обилию водорослей в реках разного типа. Поскольку бэггинг - это просто частный случай метода случайного леса, то мы можем использовать одну и ту же функцию randomForest() пакета randomForest для R. Бэггинг выполняется, если задать параметр mtry = ncol(x):

```
load(file="algae.RData") # Загрузка таблицы algae -
x <- as.data.frame(model.matrix(a1~ ., data = algae[, 1:12])[, -1])
library(randomForest)
set.seed(101)
randomForest(x, algae$a1, mtry = ncol(x))
```

Как видно из полученных результатов, прогнозирование выполнялось по 500 деревьям, в которых было использовано только 40% исходных переменных. Оценить эффективность этой модели при перекрестной проверке можно с использованием функции train() из пакета caret:

```
set.seed(101)
(bag.a1 <- train(x, algae$a1,
          preProc = c('center', 'scale'),
          method = 'rf', trControl = trainControl(method = "cv"),
          tuneGrid = expand.grid(.mtry = ncol(x))))
```

Количество деревьев B не является критическим параметром при использовании бэггинга: очень большое значение B не приведет к переобучению. На практике обычно используется значение B , достаточно большое для стабилизации ошибки величина B=100 уже обеспечивает хорошее качество предсказаний в нашем примере (по умолчанию, B=500).

```
plot(ranfor.a1$finalModel, col = "blue", lwd = 2)
plot(bag.a1$finalModel, col = "green", lwd = 2, add = TRUE)
legend("topright", c("Bagging", "RandomForrest"),
       col = c("green","blue"), lwd = 2)
```

Бустинг

Другим методом улучшения предсказаний является бустинг (boosting), идея которого заключается в итеративном процессе последовательного построения частных моделей. Каждая новая модель обучается с использованием информации об ошибках, сделанных на предыдущем этапе, а результирующая функция представляет собой линейную комбинацию всего ансамбля моделей с учетом минимизации любой штрафной функции.

В среде R для построения бустинг-моделей на основе деревьев решений можно использовать функцию gbm() из пакета gbm (Generalized Boosted Models). Процесс моделирования проходит под управлением трех гиперпараметров:

Число деревьев B (формальный параметр n.tree). В отличие от бэггинга, бустинг может, хотя и медленно, приводить к переобучению при чрезмерно большом B.

Параметр сжатия λ (shrinkage), который корректирует величину вклада каждого дополнительного дерева и контролирует скорость, с которой происходит обучение модели при реализации бустинга. Типичные значения λ варьируют от 0.01 до 0.001, и их оптимальный выбор зависит от решаемой проблемы. Для достижения хорошего качества предсказаний очень низкие значения λ требуют очень большого значения B.

Число внутренних узлов d (interaction.depth) в каждом дереве, которое контролирует сложность получаемого в результате бустинга ансамбля моделей. По своей сути, параметр d отражает глубину взаимодействий между предикторами в итоговой модели. Если эти взаимодействия не слишком выражены, то хорошо работает $d=1$, и тогда дополнительные деревья представляют собой просто “пни” (stump), т.е. содержат только один внутренний узел. В таком случае получаемый в результате бустинга ансамбль становится аддитивной моделью, поскольку каждый ее член представлен только одной переменной.

Тип решаемой задачи регулируется параметром distribution, который определяет оптимизируемую функцию:
для решения задач регрессии задается значение "gaussian" - квадратичный штраф, или "laplace" - штраф по абсолютной величине отклонения;
для задач бинарной классификации используют значение "bernoulli" - функция кросс-энтропии, или "adaboost" - экспоненциальный штраф.
Используем значение shrinkage = 0.001, установленное функцией gbm() по умолчанию. Функция summary() в отношение этого метода выводит список предикторов и соответствующие им значения показателя важности:

```
library(gbm)
set.seed(1)
xd <- cbind(a1 = algae$a1, x)
boost.a1 = gbm(a1 ~ ., data = xd, distribution = "gaussian",
n.trees = 1000, interaction.depth = 3)
summary(boost.a1, plotit = FALSE)
```

Бустинг деревьев регрессии может быть реализован также с использованием другого метода: с помощью функции bstTree() из пакета bst:

```
modelLookup("bstTree")
##     model parameter                label forReg forClass probModel
## 1 bstTree     mstop # Boosting Iterations   TRUE     TRUE     FALSE
## 2 bstTree  maxdepth        Max Tree Depth   TRUE     TRUE     FALSE
## 3 bstTree        nu             Shrinkage   TRUE     TRUE     FALSE
```

Параметры, оптимизируемые методом bstTree, имеют несколько отличающиеся названия, но фактически эквивалентный содержательный смысл. Выполним их настройку с использованием параметров, заданных по умолчанию:

```
library(bst)
(boostFit.a1 <- train(a1 ~ ., data = xd,
                method = 'bstTree', trControl = trainControl(method = "cv"),
                preProc = c('center', 'scale')))


plot(boostFit.a1)
```

Алгоритм ID3

Классификационные модели деревьев рекурсивно делят набор данных на подмножества, являющиеся все более и более гомогенными относительно определенных признаков. Создается решающее правило классификации иерархического типа и формируется ассоциативный логический ключ, дающий возможность выполнять распознавание объектов из новых выборок.

Самым простым, частным случаем деревьев решений являются деревья бинарной классификации, в узлах которых ветвление может вестись только в двух направлениях, т.е. осуществляется выбор из двух альтернатив (“да” и “нет”). Создание такой дихотомической классификационной модели осуществляется с использованием алгоритма ID3 (Interactive Dichotomizer).

В основе алгоритма (Quinlan, 1986) лежит циклическое разбиение обучающей выборки на классы в соответствии с переменной, имеющей наибольшую “классифицирующую силу”. Каждое подмножество наблюдений (объектов), выделяемое такой переменной, вновь разбивается на классы с использованием следующей переменной с наибольшей классифицирующей способностью и т.д. Разбиение заканчивается, когда в итоговом подмножестве оказываются объекты лишь одного класса. Пути движения по полученному дереву решений с верхнего уровня на самые нижние определяются логическими правилами в виде цепочек конъюнкций.

Классификационные модели деревьев рекурсивно делят набор данных на подмножества, являющиеся все более и более гомогенными относительно определенных признаков. Создается решающее правило классификации иерархического типа и формируется ассоциативный логический ключ, дающий возможность выполнять распознавание объектов из новых выборок.

Самым простым, частным случаем деревьев решений являются деревья бинарной классификации, в узлах которых ветвление может вестись только в двух направлениях, т.е. осуществляется выбор из двух альтернатив (“да” и “нет”). Создание такой дихотомической классификационной модели осуществляется с использованием алгоритма ID3 (Interactive Dichotomizer).

На языке R этот критерий может быть рассчитан следующим образом:

```
Entropy <- function(vls) {
   res <- vls/sum(vls)*log2(vls/sum(vls))
   res[vls == 0] <- 0
   -sum(res)
}
InformationGain <- function(tble) {
   entropyBefore <- Entropy(colSums(tble))
   s <- rowSums(tble)
   entropyAfter <- sum(s/sum(s) *
                       apply(tble, MARGIN = 1, FUN = Entropy ))
   informationGain <- entropyBefore - entropyAfter
   return (informationGain)
}
```

Для реализации алгоритма ID3 используем компоненты пакета data.tree: объект класса tree (дерево) и метод AddChild , добавляющий к выстраиваемому дереву очередной дочерний узел (см. http://ipub.com/wp-content):

```
IsPure <- function(data) {
   length(unique(data[, ncol(data)])) == 1
}

TrainID3 <- function(node, data) {
   node$obsCount <- nrow(data)
   #  если текущий набор данных принадлежит к одному классу, то
   if (IsPure(data)) {
      #создается лист дерева с экземплярами этого класса
      child <- node$AddChild(unique(data[, ncol(data)]))
      node$feature <- tail(names(data), 1)
      child$obsCount <- nrow(data)
      child$feature <- ''
   } else {
      # рассчитывается вектор информационных выигрышей IG
      ig <- sapply(colnames(data)[-ncol(data)],
                  function(x) InformationGain(
                     table(data[, x], data[, ncol(data)])))
      #  выбирается значение признака с наибольшей величиной IG
      feature <- names(which.max(ig))
      node$feature <- feature
      # создается подмножества данных на основе этого значения
      childObs <- split(data[, names(data) != feature,
```

```
                    drop = FALSE], data[, feature], drop = TRUE)
      #  создаются дочерние узлы дерева с именем признака
      for (i in 1:length(childObs)) {
        child <- node$AddChild(names(childObs)[i])
        # осуществляется рекурсия алгоритма для дочернего узла
        TrainID3(child, childObs[[i]])
      }
   }
}
```

Проверим теперь, насколько эффективен полученный классификатор на обучающей выборке:

```
Predict <- function(tree, features) {
   if (tree$children[[1]]$isLeaf)
      return(tree$children[[1]]$name)
   child <- tree$children[[features[[tree$feature]]]]
   return( Predict(child, features))
}
pred <- apply(DFaceN[, -17], 1, function(x) Predict(tree, x))
table(Факт = DFaceN$Группа, Прогноз = pred)
##          Прогноз
## Факт      Демократ Патриот
##   Демократ        8       0
##   Патриот         0       8
```

Предсказание электоральной группы по всей обучающей выборке, как и в случае логистической регрессии, также оказалось безошибочным. Однако нет уверенности, что построенные деревья будут столь же успешны при практической реализации на “свежих” данных. Для оценки эффективности модели выполним перекрестную проверку:

```
Nerr <- 0
for (i in 1:nrow(DFaceN)) {
   tree <- Node$new("DFaceN")
   TrainID3(tree, DFaceN[-i, ])
   Nerr = Nerr +
      sum(DFaceN[i,17] != Predict(tree,DFaceN[i, -17]))  }
(Nerr/nrow(DFaceN))
```

Варианты заданий

Таблица с перечнем из номенклатурного списка супермаркета «Новая Эра»

Код	Наименование	Категория	Эталонная цена	
001	Кока-Кола	Напитки	60	
002	Спрайт	Напитки	70	
003	Добрый Палпи	Напитки	50	
004	Сила Фруктов	Напитки	45	
005	Кофе Лебо Голд		250	
006	Чай Гринфилд		110	
007	Чай Липтон		115	
008	Пиво Хейнекен		80	
009	Пиво Холсен		90	
010	Пиво Балтика		60	
011	Чипсы Лэйс		70	
012	Чипсы Эстрелла		83	
013	Чипсы Принглс		160	
	Пиво Бад		75	
014	Лимонад Аквалайн		44	
015	Лимонад Лайман		47	
016	Сок Вико		70	
017	Сок Джей 7		110	
018	Пюре Агуша		60	
019	Грудинка Столичная		140	
020	Балык Дарницкий		154	
021	Колбаса Краковская		52	
022	Сервелат Старорузский		200	
023	Колбаса Докторская		350	
024	Ветчина Мясная		165	

025	Сосиски Папа Может		125	
026	Сосиски Премиум		140	
027	Йогурт Вкуснотеево		85	
028	Ряженка Домик в Деревне		84	
030	Сливки Простоквашина		126	
031	Йогурт Чудо Детки		42	
032	Йогурт Данон		35	
033	Творожок Чудо		83	
034	Пломбир Ванильный		45	
035	Мороженое 48 копеек		81	
036	Сыр Брест Литовск		176	
037	Сыр Чеддер		186	
038	Майонез Провансаль Оливковый	Продукты	67	
039	Майонез Провансаль Ряба	Продукты	73	
040	Пицца Ресторанте	Продукты	264	
041	Пельмени Мяснушки	Продукты	127	
042	Кофе Черная Карта		300	
043	Чай Акбар		76	
044	Кофе Амбассадор		231	
045	Кофе Милагро		240	
046	Чай Ахмад	Деликатесы	280	
047	Кофе Якобс	Деликатесы	386	
048	Чай Тесс	Деликатесы	88	
049	Конфеты Вдох-	Деликатесы	250	

	новение			
050	Сайра Тихо-океанская	Морепродукты	98	

Каждый пункт номенклатурного списка характеризуется коэффициентом дня недели (от понедельника до воскресенья), сезонным коэффициентом времени года и температурным коэффициентом. Кп- коэф. понедельника, Кв – коэф. вторника, Кс – к-т среды, Кч – к-т четверга, Кп- к-т пятницы, Ксб- к-т субботы, Квс – к-т воскресенья, Кз – зимний коэфф., Кл – летний коэф., Ко- к-т осени, Кт – температурный коэффициент. Эталонная цена умножается на характеристические коэффициенты и так рассчитывается значение скидки на товар в выбранный день недели. В задании лабораторной работы надо классифицировать полученную скидку выбранных товаров в определенный день по категориям: «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70%.

Вариант 1

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
001	1							0.7	0.54	1	1	0.45	
002	0.77											1	
003													
004													
005													
006													
007													
008												0.15	
009	1		1				1			1		0.23	
010	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 3

2).Для генетического алгоритма выбрать: генную бинарную комбинацию **1001101**

3).Для алгоритма CART в функции rpart выбрать параметр method = "poisson"
4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 300, в функции train выбрать параметр method = "bagFDA"
5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 400, параметр distribution = “laplace”, параметр bag.Fraction = 0.5
6). Результаты визуализировать и сравнить.

Вариант 2

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
011	1							0.7	0.54	1	1	0.45	
012	0.77											1	
013													
014													
015													
016													
017													
018												0.15	
019	1		1				1			1		0.23	
020	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 4
2).Для генетического алгоритма выбрать: генную бинарную комбинацию **1101101**
3).Для алгоритма CART в функции rpart выбрать параметр method = "anova"
4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 400, в функции train выбрать параметр method = "Adabag"
5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 300, параметр distribution = “bernoully”, параметр bag.Fraction = 0.47
6). Результаты визуализировать и сравнить.

Вариант 3

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
021	1							0.7	0.54	1	1	0.45	
022	0.77											1	
023													
024													
025													
026													
027													
028												0.15	
029	1		1				1			1		0.23	
030	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 5

2).Для генетического алгоритма выбрать: генную бинарную комбинацию **1101000**

3).Для алгоритма CART в функции rpart выбрать параметр method = "class"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 600, в функции train выбрать параметр method = "Treebag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 460, параметр distribution = “gussian”, параметр bag.Fraction = 0.65

6). Результаты визуализировать и сравнить.

Вариант 4

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
031	1							0.7	0.54	1	1	0.45	
032	0.77											1	
033													

034													
035													
036													
037													
038												0.15	
039	1		1				1			1		0.23	
040	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 6

2).Для генетического алгоритма выбрать: генную бинарную комбинацию **1011001**

3).Для алгоритма CART в функции rpart выбрать параметр method = "exp"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 280, в функции train выбрать параметр method = "Logicbag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 380, параметр distribution = “adaBoost”, параметр bag.Fraction = 0.39

6). Результаты визуализировать и сравнить.

Вариант 5

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
041	1							0.7	0.54	1	1	0.45	
042	0.77											1	
043													
044													
045													
046													
047													
048												0.15	
049	1		1				1			1		0.23	
050	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 7

2).Для генетического алгоритма выбрать: генную бинарную комбинацию **1011101**

3).Для алгоритма CART в функции rpart выбрать параметр method = "poisson"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 580, в функции train выбрать параметр method = "bagEarth"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 630, параметр distribution = “coxPH”, параметр bag.Fraction = 0.88

6). Результаты визуализировать и сравнить.

Вариант 6

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
011	1							0.7	0.54	1	1	0.45	
013	0.77											1	
015													
017													
019													
021													
023													
025												0.15	
027	1		1				1			1		0.23	
029	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций Min-Num= 5

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 0**011100**

3).Для алгоритма CART в функции rpart выбрать параметр method = "anova"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 430, в функции train выбрать параметр method = "Adabag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 720, параметр distribution = “poisson”, параметр bag.Fraction = 0.72

6). Результаты визуализировать и сравнить.

Вариант 7

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 8

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 1**010101**

3).Для алгоритма CART в функции rpart выбрать параметр method = "class"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 700, в функции train выбрать параметр method = "Treebag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 340, параметр distribution = “bernaully”, параметр bag.Fraction = 0.81

6). Результаты визуализировать и сравнить.

Вариант 8

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 5

2).Для генетического алгоритма выбрать: генную бинарную комбинацию **11**10111

3).Для алгоритма CART в функции rpart выбрать параметр method = "anova"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 730, в функции train выбрать параметр method = "adabag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 590, параметр distribution = “laplace”, параметр bag.Fraction = 0.48

6). Результаты визуализировать и сравнить.

Вариант 9

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													

035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 4

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 0**011101**

3).Для алгоритма CART в функции rpart выбрать параметр method = "class"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 620, в функции train выбрать параметр method = "Treebag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 540, параметр distribution = “adaboost”, параметр bag.Fraction = 0.79

6). Результаты визуализировать и сравнить.

Вариант 10

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 8

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 11**10101**
3).Для алгоритма CART в функции rpart выбрать параметр method = "poisson"
4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 300, в функции train выбрать параметр method = "adabag"
5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 380, параметр distribution = “bernaully”, параметр bag.Fraction = 0.55
6). Результаты визуализировать и сравнить.

Вариант 11

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций Min-Num= 8
2).Для генетического алгоритма выбрать: генную бинарную комбинацию 0**010100**
3).Для алгоритма CART в функции rpart выбрать параметр method = "anova"
4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 780, в функции train выбрать параметр method = "Treebag"
5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 340, параметр distribution = “bernaully”, параметр bag.Fraction = 0.81
6). Результаты визуализировать и сравнить.

Вариант 12

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035	1												
040													
045													
050	1	0.73											
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 8

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 11**10100**

3).Для алгоритма CART в функции rpart выбрать параметр method = "poisson"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 880, в функции train выбрать параметр method = "Treebag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 340, параметр distribution = “bernaully”, параметр bag.Fraction = 0.81

6). Результаты визуализировать и сравнить.

Вариант 13

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	
030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций MinNum= 5

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 010**0101**

3).Для алгоритма CART в функции rpart выбрать параметр method = "exp"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 710, в функции train выбрать параметр method = "Treebag"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 340, параметр distribution = “coxph”, параметр bag.Fraction = 0.61

6). Результаты визуализировать и сравнить.

Вариант 14

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
020	1							0.7	0.54	1	1	0.45	
025	0.77											1	

030													
035													
040													
045													
050													
001												0.15	
003	1		1				1			1		0.23	
011	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций Min-Num= 7

2).Для генетического алгоритма выбрать: генную бинарную комбинацию 1**010111**

3).Для алгоритма CART в функции rpart выбрать параметр method = "class"

4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 624, в функции train выбрать параметр method = "bagFDA"

5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 340, параметр distribution = “bernoully”, параметр bag.Fraction = 0.441

6). Результаты визуализировать и сравнить.

Вариант 15

Разработать логический классификатор с использованием алгоритмов **«Кора», «ID3», «CART», «Бэггинг», «Бустинг», генетического алгоритма** для классификации товаров супермаркета по категориям «Скидки - Нет» 0-4 %, «Скидка-мини» 5-25 %, «Выгодная Скидка» 26-40%, «Супер Скидка» 50-70% за пять дней.

Код	Кп	Кв	Кс	Кч	Кп	Ксб	Квс	Кз	Квн	Кл	Ко	Кт	
050	1	1	0.9	0.7				0.7	0.54	1	1	0.45	
030	0.77											1	
034		0.1											
021													
014	1		1			1			1			1	
046													
036													
049												0.15	
009	1		1				1			1		0.23	
014	0.58												

1).Для алгоритма Кора: выбрать частоту встречаемости конъюнкций Min-Num= 8
2).Для генетического алгоритма выбрать: генную бинарную комбинацию 11**10001**
3).Для алгоритма CART в функции rpart выбрать параметр method = "exp"
4).Для алгоритма Bagging в функции RandomForest выбрать параметр N.trees= 200, в функции train выбрать параметр method = "bagEarth"
5).Для алгоритма Boosting в функции gbm выбрать параметр N.trees = 510, параметр distribution = “laplace”, параметр bag.Fraction = 0.49
6). Результаты визуализировать и сравнить.