Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого
—
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта

ЛАБОРАТОРНАЯ РАБОТА №4

«Деревья решений»

по дисциплине «Машинное обучение, часть1»

Выполнил: студент группы
3540201/20302 С.А. Ляхова

<*подпись*>

Проверил: Л.В. Уткин
д.т.н., профессор

<*подпись*>

Санкт-Петербург
2022

Содержание

1. Цель работы

Исследовать пакет tree языка R, формирующий и оптимизирующий деревья решений, выполнив поставленные задачи и проанализировав результаты

2. Формулировка задания

1) Загрузите набор данных Glass из пакета “mlbench”. Набор данных (признаки, классы) был изучен в работе «Метод ближайших соседей». Постройте дерево классификации для модели, задаваемой следующей формулой: Type~ ., дайте интерпретацию полученным результатам. При рисовании дерева используйте параметр cex=0.7 для уменьшения размера текста на рисунке, например, text(bc.tr,cex=0.7) или draw.tree(bc.tr,cex=0.7). Является ли построенное дерево избыточным? Выполните все операции оптимизации дерева.

2) Загрузите набор данных spam7 из пакета DAAG. Постройте дерево классификации для модели, задаваемой следующей формулой: yesno ~., дайте интерпретацию полученным результатам. Запустите процедуру “cost-complexity prunning” с выбором параметра k по умолчанию, method = ’misclass’, выведите полученную последовательность деревьев. Какое из полученных деревьев, на Ваш взгляд, является оптимальным? Объясните свой выбор.

3) Загрузите набор данных nsw74psid1 из пакета DAAG. Постройте регрессионное дерево для модели, задаваемой следующей формулой: re78~.. Постройте регрессионную модель и SVM-регрессию для данной формулы. Сравните качество построенных моделей, выберите оптимальную модель и объясните свой выбор.

4) Загрузите набор данных Lenses Data Set из файла Lenses.txt:

3 класса (последний столбец):

1 : пациенту следует носить жесткие контактные линзы,

2 : пациенту следует носить мягкие контактные линзы,

3 : пациенту не следует носить контактные линзы.

Признаки (категориальные):

1. возраст пациента: (1) молодой, (2) предстарческая дальнозоркость, (3) старческая дальнозоркость

2. состояние зрения: (1) близорукий, (2) дальнозоркий

3. астигматизм: (1) нет, (2) да

4. состояние слезы: (1) сокращенная, (2) нормальная

Постройте дерево решений. Какие линзы надо носить при предстарческой дальнозоркости, близорукости, при наличии астигматизма и сокращенной слезы?

5) Постройте дерево решений для обучающего множества Glass, данные которого характеризуются 10-ю признаками:

1. Id number: 1 to 214; 2. RI: показатель преломления; 3. Na: сода (процент содержания в соотвествующем оксиде); 4. Mg; 5. Al; 6. Si; 7. K; 8. Ca; 9. Ba; 10. Fe.

Классы характеризуют тип стекла:

(1) окна зданий, плавильная обработка

(2) окна зданий, не плавильная обработка

(3) автомобильные окна, плавильная обработка

(4) автомобильные окна, не плавильная обработка (нет в базе)

(5) контейнеры

(6) посуда

(7) фары

Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки. Это выполняется командой glass <- glass[,-1].

Определите, к какому типу стекла относится экземпляр с характеристиками

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

6) Для построения классификатора используйте заранее сгенерированные обучающие и тестовые выборки, хранящиеся в файлах svmdata4.txt, svmdata4test.txt.

7) Разработать классификатор на основе дерева решений 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

3. Ход работы

Задание №1

Построенное на рисунке 1 дерево без оптимизаций является избыточным, следовательно нуждается в оптимизации.

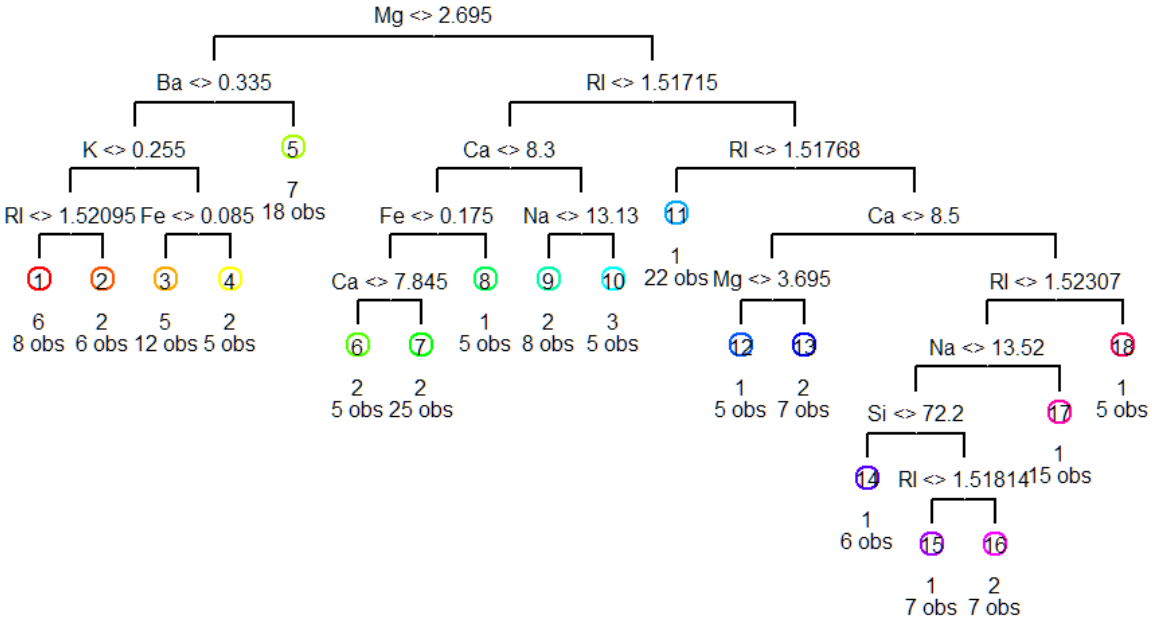

Рисунок 1. Дерево решений для набора данных Glass до оптимизации

Дерево без оптимизации имеет результат 73.5%.

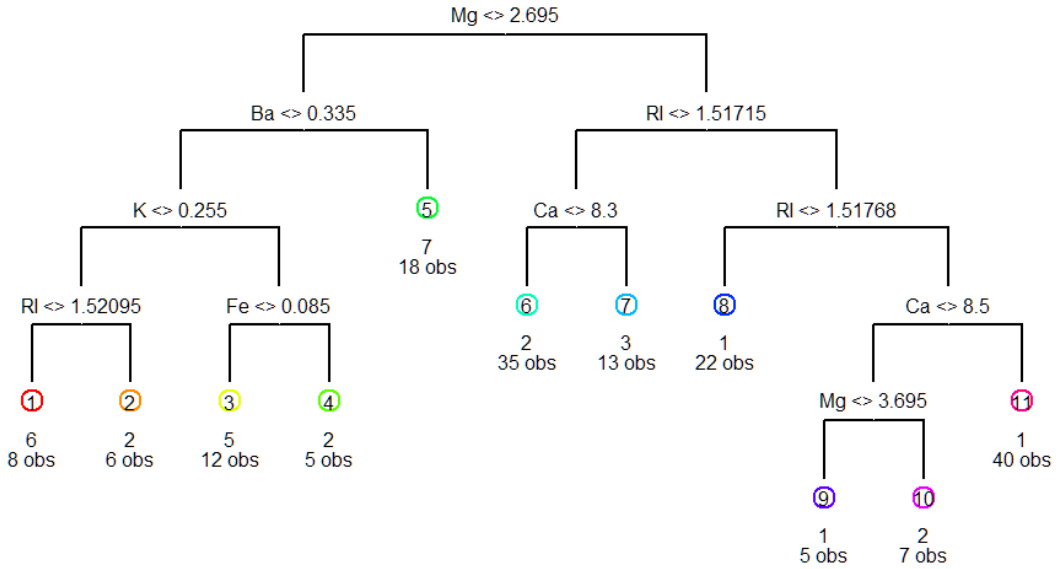

Рисунок 2. Дерево решений для набора данных Glass после применения метода prune

Дерево после применения метода prune имеет результат 71.4%.

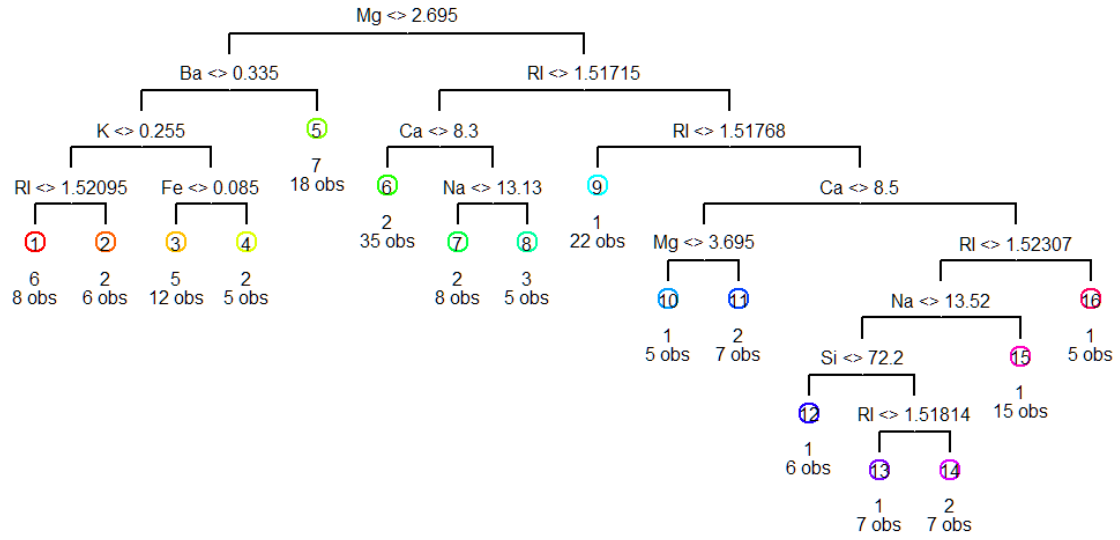

Рисунок 3. Дерево решений для набора данных Glass после применения метода snip

Дерево после применения метода snip имеет результат 74.2%.

Получившееся на рисунке 2 дерево лишено всех наименее значимых вершин и не является избыточным.

В данном примере получилось, что дерево без оптимизации работает лучше, чем при использовании метода оптимизации prune, но уступает методу snip, с удаленным 12 узлом. Можно заметить, что дерево решений после оптимизации имеет меньше узлов, что делает его более простым для интерпретации.

Задание №2

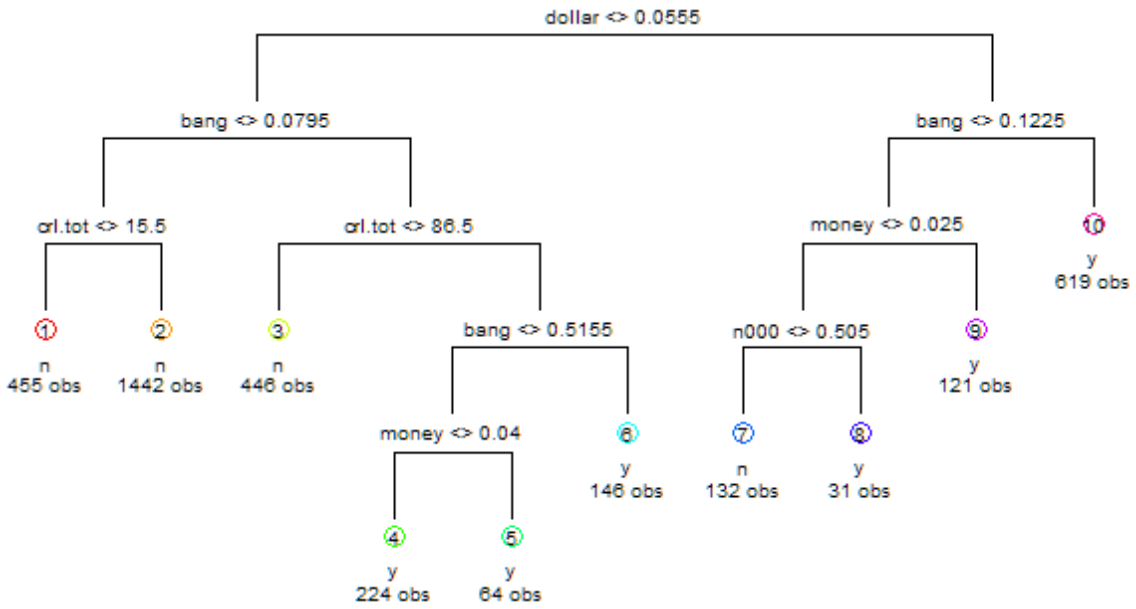

Рисунок 4. Дерево решений для данных spam без оптимизации

Дерево решений для данных spam без оптимизации имеет результат 89.3%.

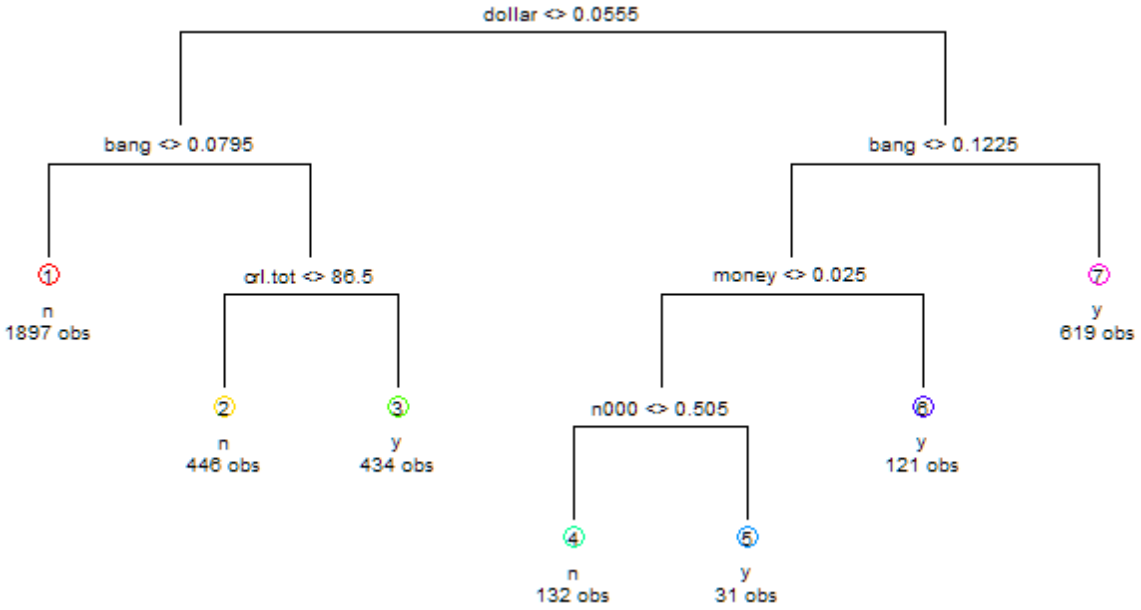

Рисунок 5. Дерево решений для данных spam с оптимизацией

Дерево решений для данных spam с оптимизацией имеет результат 89.2%.

В результате оптимизации дерева решений можем заметить, что листья 1, 2, 4, 5, 6 (рисунок 4) были отсечены (объединены в один лист, рисунок 5). Это можно объяснить тем, что значения в этом поддереве принадлежали одному классу. Получаем, что после оптимизации дерево перестало быть избыточным. Но при этом результат работы модели не отличаются.

Задание №3

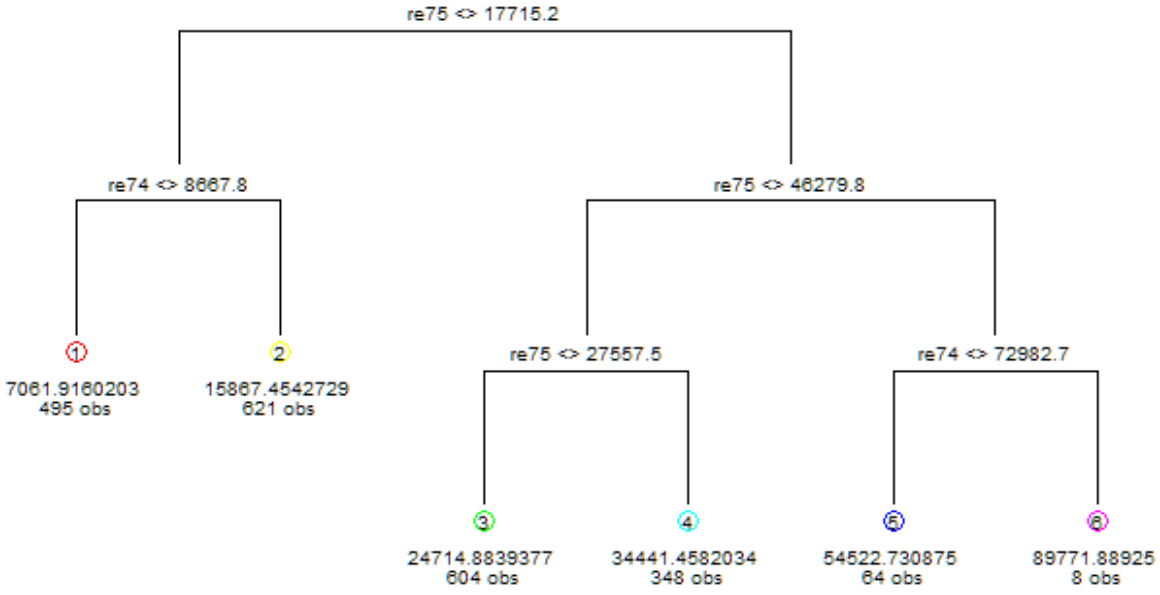

Рисунок 6. Регрессионное дерево для датасета nsw74psid1

На рисунке 6 показано дерево решений для регрессии.

Для данной формулы также была построена SVM-регрессия. Модели были построены на основе обучающей выборки (80% от общего объема данных). Чтобы сравнить построенное дерево решений с SVM, подсчитаем MSE (среднеквадратичное отклонение): для дерева это значение составило 1.214768e^(+14), а для метода опорных векторов - 1.092517e^(+14).

Получили, что SVM работает лучше дерева решений.

Задание №4

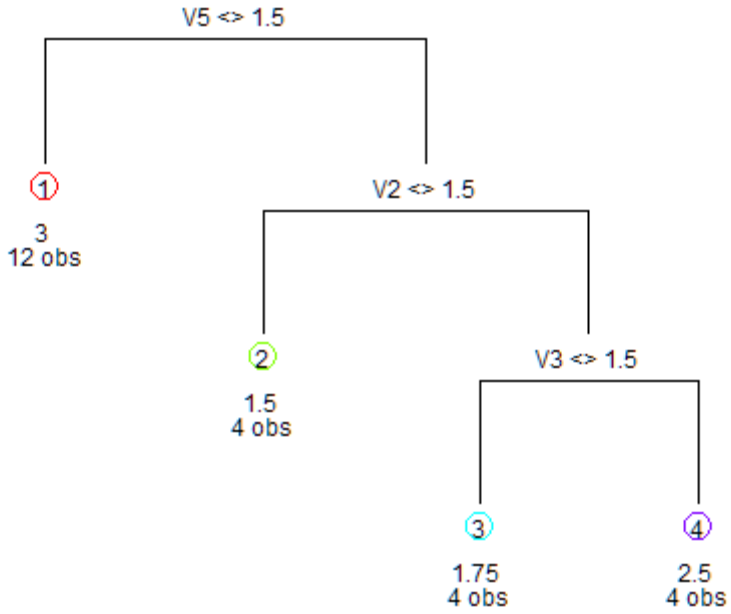

Рисунок 7. Дерево решений для датасета Lenses

Дерево решений, построенное в ходе анализа датасета, представлено на рисунке 7. С помощью функции predict установлено, что при предстарческой дальнозоркости, близорукости, при наличии астигматизма и сокращенной слезы: 3. «пациенту не следует носить контактные линзы».

Задание №5

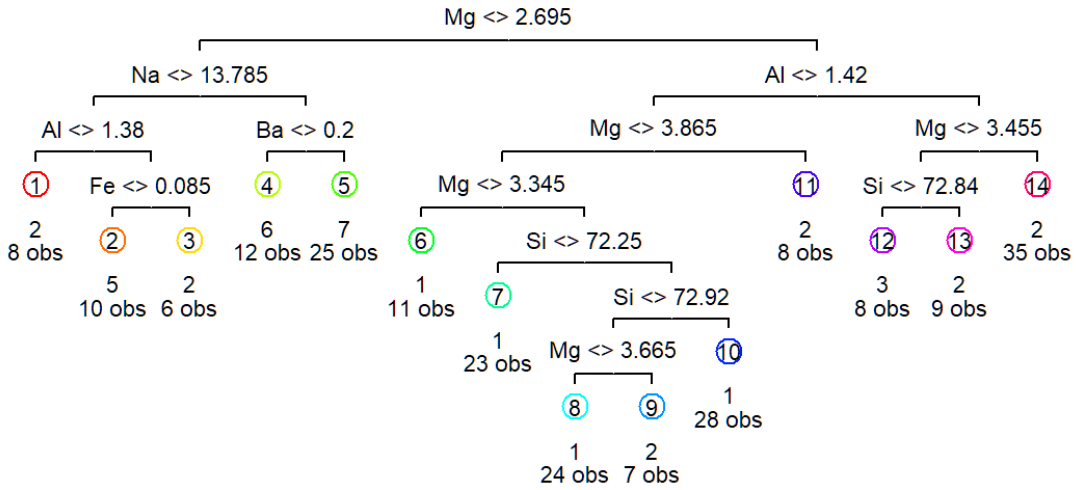

Рисунок 8. Дерево решений для датасета Glass

В результате анализа датасета было построено дерево решений (рисунок 8). Построенный классификатор с вероятностью 87.5% отнес представленный пример к классу (2): окна зданий, не плавильная обработка. Результат не совпал с результатом Лабораторной №2.

Задание №6

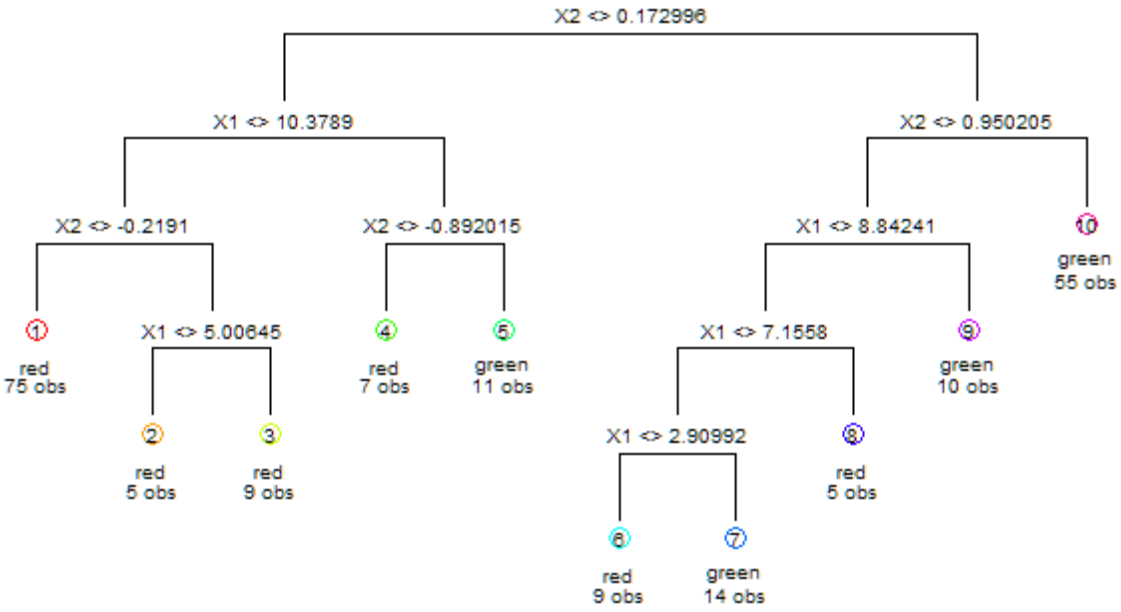

Рисунок 9. Дерево решений датасета svmdata4

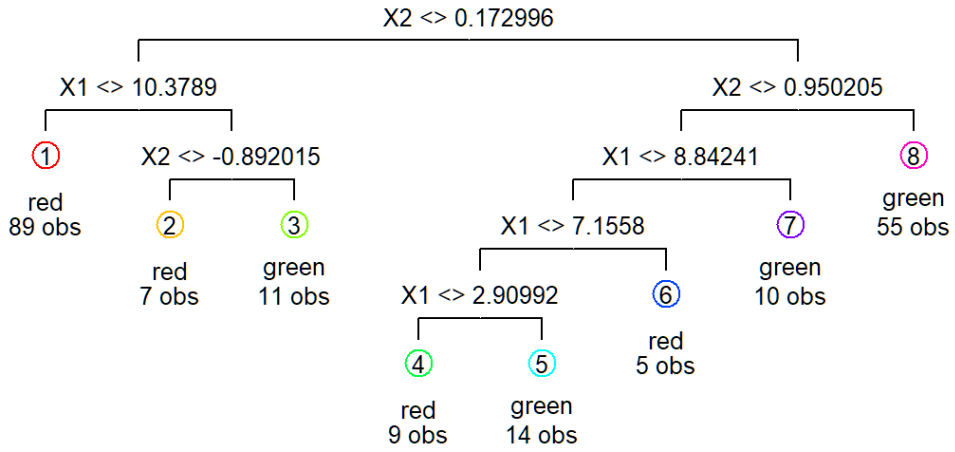

Рисунок 10. Дерево решений датасета svmdata4 с оптимизацией

Было построено дерево решений для датасета svmdata4. Данное дерево показывает точность классификации 89.8%.

Задание №7

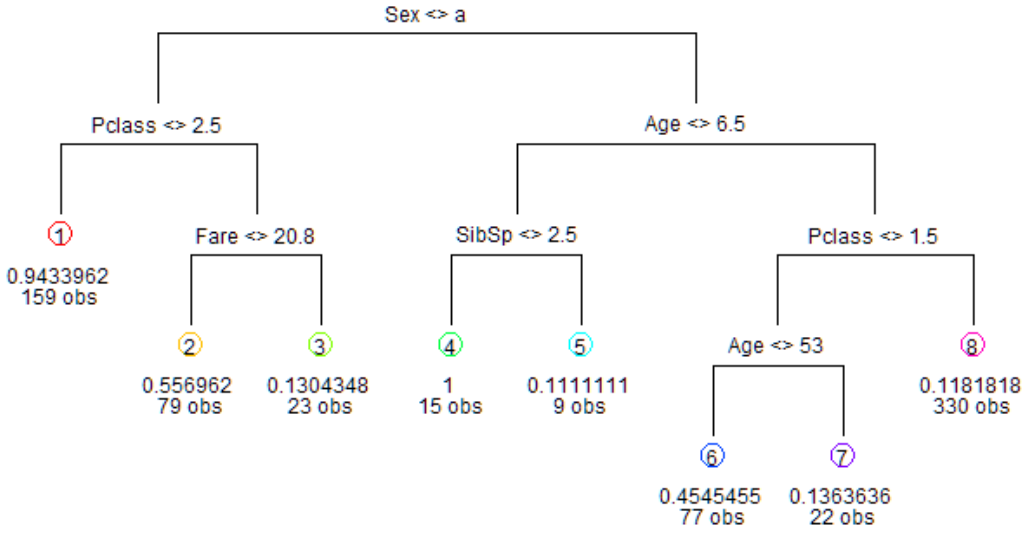

Рисунок 11. Результаты построения дерева решений датасета Titanic

Было построено дерево решений датасета Titanic (рисунок 11). Реальное соотношение умерших и выживших в выборке: 61.61% умерших, 38.39% выживших.

Дерево решений показывает результат 38.38% выживших и 61.62% умерших. Таким образом, погрешность очень мала (0.01%).

4. Вывод

В ходе работы был рассмотрен метод деревьев решений как для задач классификации, так и для задач регрессии. Этот метод машинного обучения показывает хорошие результаты, однако имеет ряд недостатков.

Деревья решений очень легко интерпретировать в сравнении с другими алгоритмами машинного обучения, однако они чувствительны к шумам и склонны к переобучению, что требует тщательного выбора значений параметров.

Необходимо правильно выполнять оптимизацию дерева, чтобы избегать проблем переобучения. В языке R ограничения на вложенность дерева 32, что не позволяет строить классификатор для данных большой размерности.

При этом этот метод в совокупности с другими алгоритмами машинного обучения позволяет стоить классификаторы с высокой точностью.

Приложение 1

```
# Задание 1 ############################################################
data("Glass")

ratio <- 0.8
n <- nrow(Glass)
nt <- as.integer(n * ratio)

glass_rand <- Glass[order(runif(n)), ]
glass_train <- glass_rand[1: nt, ]
glass_test <- glass_rand[(nt + 1): n, ]

#------------------
tree1 <- tree(Type ~ ., data = glass_train)

png(paste(path, "glass_tree.png"), width = 720, height = 480)
draw.tree(tree1, cex = 0.5)
dev.off()

pred1 <- predict(tree1, glass_test)

s <- vector()

for(i in 1:nrow(pred1))
{
  s <- append(s, mse(1, pred1[i, glass_test$Type[i]]))
}

1 - mean(s)

#------------------
tree2 <- prune.tree(tree1, 10)

png(paste(path, "glass_tree_optim.png"), width = 720, height = 480)
draw.tree(tree2, cex=0.5)
dev.off()

pred2 <- predict(tree2, glass_test)

s <- vector()

for(i in 1:nrow(pred2))
{
  s <- append(s, mse(1, pred2[i, glass_test$Type[i]]))
}

1 - mean(s)

#------------------
```

```
tree3 <- snip.tree(tree1, nodes = c(12))

png(paste(path, "glass_tree_snip.png"), width = 720, height = 480)
draw.tree(tree3, cex=0.5)
dev.off()

pred3 <- predict(tree3, glass_test)

s <- vector()

for(i in 1:nrow(pred3))
{
  s <- append(s, mse(1, pred3[i, glass_test$Type[i]]))
}

1 - mean(s)
```

Приложение 2

```
# Задание 2 ############################################################
library(DAAG)
data(spam7)

ratio <- 0.8
n <- nrow(spam7)
nt <- as.integer(n * ratio)

spam_rand <- spam7[order(runif(n)), ]
spam_train <- spam_rand[1: nt, ]
spam_test <- spam_rand[(nt + 1): n, ]

tree1 <- tree(yesno ~ ., data = spam_train)
png(paste(path, "spam_tree.png"), width = 720, height = 480)
draw.tree(tree1, cex=0.8)
dev.off()

pred1 <- predict(tree1, spam_test)

s <- vector()

for(i in 1:nrow(pred1))
{
  s <- append(s, mse(1, pred1[i, spam_test$yesno[i]]))
}

1-mean(s)

#--------------
tree2 <- prune.tree(tree1, method = "misclass", k=1)
png(paste(path, "spam_tree_prune.png"), width = 720, height = 480)
draw.tree(tree2, cex=0.8)
dev.off()

pred2 <- predict(tree2, spam_test)

s <- vector()

for(i in 1:nrow(pred2))
{
  s <- append(s, mse(1, pred2[i, spam_test$yesno[i]]))
}
1-mean(s)
prune.tree(tree1, method = "misclass")$k
draw.tree(prune.tree(tree1, k = 0), cex=0.7)
draw.tree(prune.tree(tree1, k = 11), cex=0.7)
draw.tree(prune.tree(tree1, k = 105), cex=0.7)
draw.tree(prune.tree(tree1, k = 685), cex=0.7)
```

Приложение 3

```
# Задание 3 ############################################################
library(e1071)

data(nsw74psid1)

n <- nrow(nsw74psid1)

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

data_rnd <- nsw74psid1[order(runif(n)), ]
nsw_train <- data_rnd[1:nt, ]
nsw_test <- data_rnd[(nt+1):n, ]

tree_nsw <- tree(re78 ~ ., nsw_train)

png(paste(path, "nsw_tree.png"), width = 720, height = 480)
draw.tree(tree_nsw, cex = 0.8)
dev.off()

pred1 <- predict(tree_nsw, nsw_test[, !colnames(nsw_test) %in% c("re78")])

mean(mse(sum(pred1), nsw_test$re78))

svm_nsw <- svm(re78 ~ ., nsw_train, type = "eps-regression")

pred2 <- predict(svm_nsw, nsw_test[, !colnames(nsw_test) %in% c("re78")])

mean(mse(sum(pred2), nsw_test$re78))
```

Приложение 4

```
# Задание 4 ############################################################

library(mlbench)
library(maptree)

Lenses <- read.table(paste(path, "Lenses.txt", sep = ""), header = FALSE)
Lenses$V1 <- NULL

len_tree1 <- tree.control(nrow(Lenses), mincut = 2, minsize = 6)
len_tree2 <- tree(V6 ~ ., Lenses, control = len_tree1)

png(paste(path, "lenses.png"))
draw.tree(len_tree2)
dev.off()

print(predict(len_tree2, data.frame(V2 = 2, V3 = 1, V4 = 2, V5 = 1)))
```

Приложение 5

```
# Задание 5 ############################################################

data(Glass)

glass <- Glass[, -1]

glass_tree <- tree(Type ~ ., Glass)
glass_tree1 <- prune.tree(glass_tree, k = 10)

png(paste(path, "glass_predict.png"), width = 720, height = 480)
draw.tree(glass_tree1, cex = 0.7)
dev.off()

predict(
  glass_tree1,
  data.frame(
    RI = 1.516,
    Na = 11.7,
    Mg = 1.01,
    Al = 1.19,
    Si = 72.59,
    K = 0.43,
    Ca = 11.44,
    Ba = 0.02,
    Fe = 0.1))
```

Приложение 6

```
# Задание 6 ############################################################

svm_train <- read.table(paste(path, "svmdata4.txt", sep = ""), stringsAsFactors = TRUE)
svm_test <- read.table(paste(path, "svmdata4test.txt", sep = ""), stringsAsFactors = TRUE)

tree1 <- tree(Colors ~ ., data = svm_train)

png(paste(path, "svmdata4.png"), width = 720, height = 480)
draw.tree(tree1, cex=0.8)
dev.off()

pred1 <- predict(tree1, svm_test)

s <- vector()

for(i in 1:nrow(pred1)){
  s <- append(s, mse(1, pred1[i, svm_test$Colors[i]]))
}

1 - mean(s)

tree2 <- prune.tree(tree1, 10)

png(paste(path, "svmdata4_optim.png"), width = 720, height = 480)
draw.tree(tree2, cex=0.8)
dev.off()

pred1 <- predict(tree2, svm_test)

s <- vector()

for(i in 1:nrow(pred1)){
  s <- append(s, mse(1, pred1[i, svm_test$Colors[i]]))
}

1 - mean(s)
```

Приложение 7

```
# Задание 7 ############################################################

T_train <- read.csv(paste(path, "Titanic_train.csv", sep = ""), stringsAsFactors = TRUE)
T_test <- read.csv(paste(path, "Titanic_test.csv", sep = ""),  stringsAsFactors = TRUE)

T_train$Name <- paste(T_train$Name, sep = ",", T_train$X)
T_train <- T_train[,-5]

T_test$Name <- paste(T_test$Name, sep = ",", T_test$X)
T_test <- T_test[,-4]

null_names <- c("PassengerId", "Name", "Ticket", "Cabin")

T_train[, colnames(T_train) %in% null_names] <- NULL
T_test[, colnames(T_test) %in% null_names] <- NULL

nt <- nrow(T_train)
n <- nt + nrow(T_test)

T_classifier <- tree(Survived ~ ., data = T_train)

png(paste(path, "Titanic.png"), width = 720, height = 480)
draw.tree(T_classifier)
dev.off()

T_predicted <- predict(T_classifier, T_test)

survived <- length(T_predicted[T_predicted > 0.5]) / length(T_predicted)
dead <- length(T_predicted[T_predicted <= 0.5]) / length(T_predicted)
survived
dead

prop.table(table(T_train$Survived))
```