Лабораторная работа №4 «Деревья решений»

Работу выполнила студентка группы 5140201/30301 Фазылова Алика

Задание 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.

Решение

Построим дерево решений (рисунок 1)

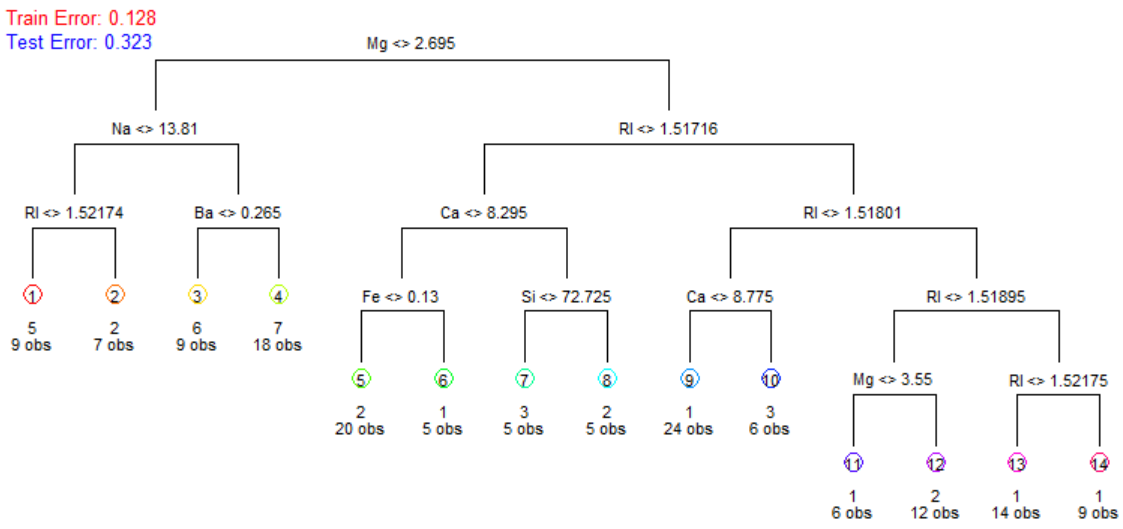

Рисунок 1 – Дерево решений

Да, дерево является избыточным, то есть оно подстраивается под параметры обучающей выборки, что может привести к переобучению дерева. На данный момент ошибка на тренировочной выборке составляет 0.128, а на тестовой 0.323. Выставим параметр k=10 в prune.tree, количество узлов при этом уменьшилось (рисунок 2).

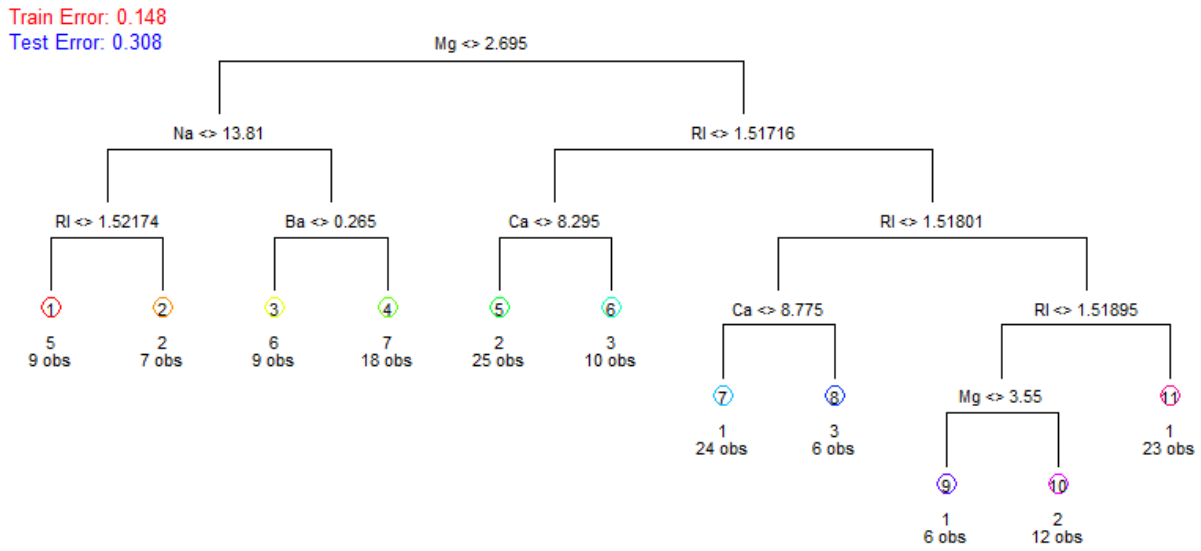

Рисунок 2 – Дерево решений при k=10

После уменьшения количества допустимых узлов, избыточность дерева уменьшилась, и ошибка на тестовой выборке снизилась до 0.308, что говорит о повышении обобщающей способности модели.

Уменьшим еще избыточность, “обрезав” построенное дерево в узле с номером 30.

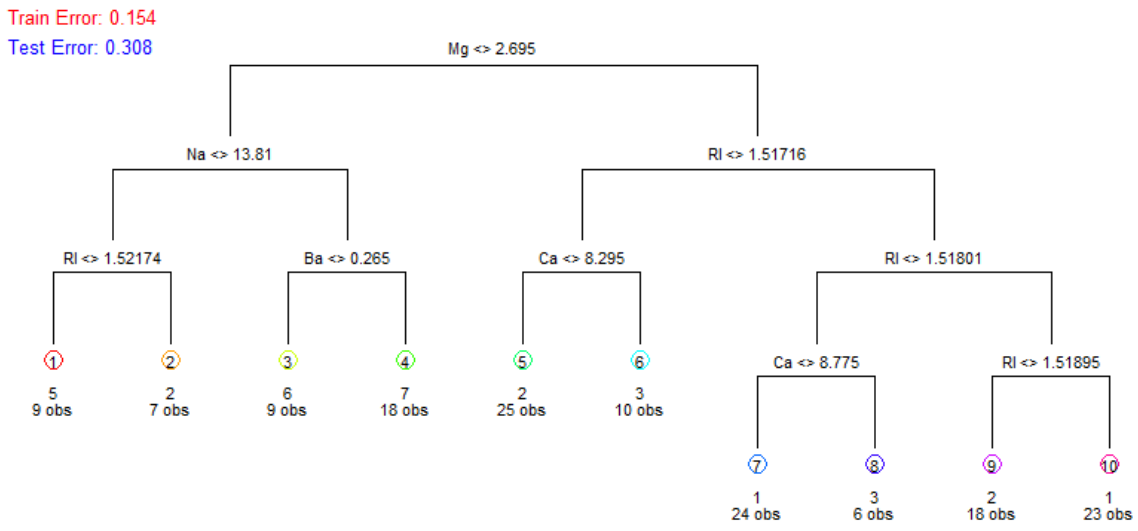

Рисунок 3 – Дерево решений оптимальное

Ошибка на тестовой выборке осталась прежней, а на тренировочной возросла. Таким образом, мы смогли убрать избыточность дерева и повысить гибкость модели.

Листинг кода 1 задачи:

```
library(tree)
# Загрузка данных Glass из пакета mlbench
data(Glass)
glass_data <- Glass
# Функция для подсчета ошибок и построения графика
plot_and_model_error <- function(model) {
  # Предсказания
  train_predict <- predict(model, train_data, type = "class")
  test_predict <- predict(model, test_data, type = "class")

  # Оценка ошибок
  train_error <- sum(train_predict != train_data$Type) / nrow(train_data)
  test_error <- sum(test_predict != test_data$Type) / nrow(test_data)
  # Построение графика
  draw.tree(model, cex = 0.7)
  text(0.5, 0.3, paste("Train Error:", round(train_error, 3)), col = "red", cex = 0.8, pos = 4)
  text(0.5, 0.01, paste("Test Error:", round(test_error, 3)), col = "blue", cex = 0.8, pos = 4)
}

#значение рандома
set.seed(1235)

# Разделение данных на обучающую и тестовую выборки
train_indices <- sample(1:nrow(glass_data), 0.7 * nrow(glass_data))
train_data <- glass_data[train_indices, ]
test_data <- glass_data[-train_indices, ]

# Построение дерева классификации на обучающей выборке
glass_tree <- tree(Type ~ ., data = train_data)

# Оценка модели вывод графика
plot_and_model_error(glass_tree)

#отсечение k=10
glass_tree1 <- prune.tree(glass_tree, k = 10)

# Оценка модели вывод графика после уменьшения кол-ва узлов
plot_and_model_error(glass_tree1)

#Узел Ri<1.51895 избыточный посмотрим номер узла
glass_tree1

#обрежем дерево в 30 узле
glass_tree2 <- snip.tree(glass_tree1, nodes = 30)

# Оценка модели вывод графика после обрезки в 30 узле
plot_and_model_error(glass_tree2)
```

Задание 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.

Решение

Найдем граничные k при котором количество узлов будет изменятся.

k= 0; 2.5; 97.

Построим для них деревья (рисунки 4-6)

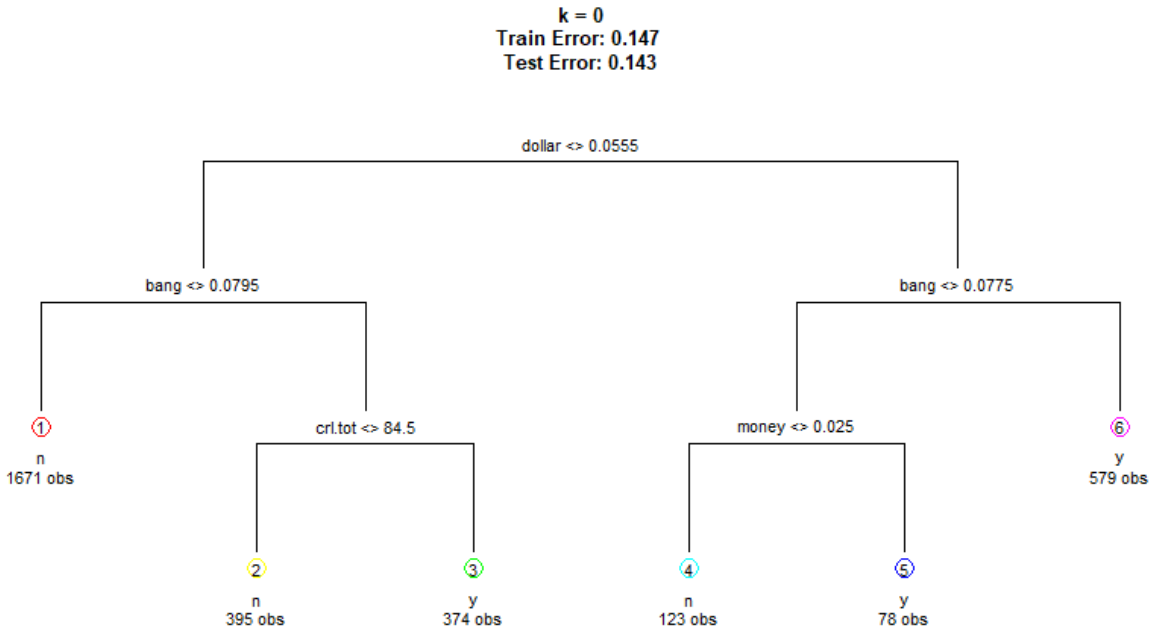

Рисунок 4 – Дерево решений k=0

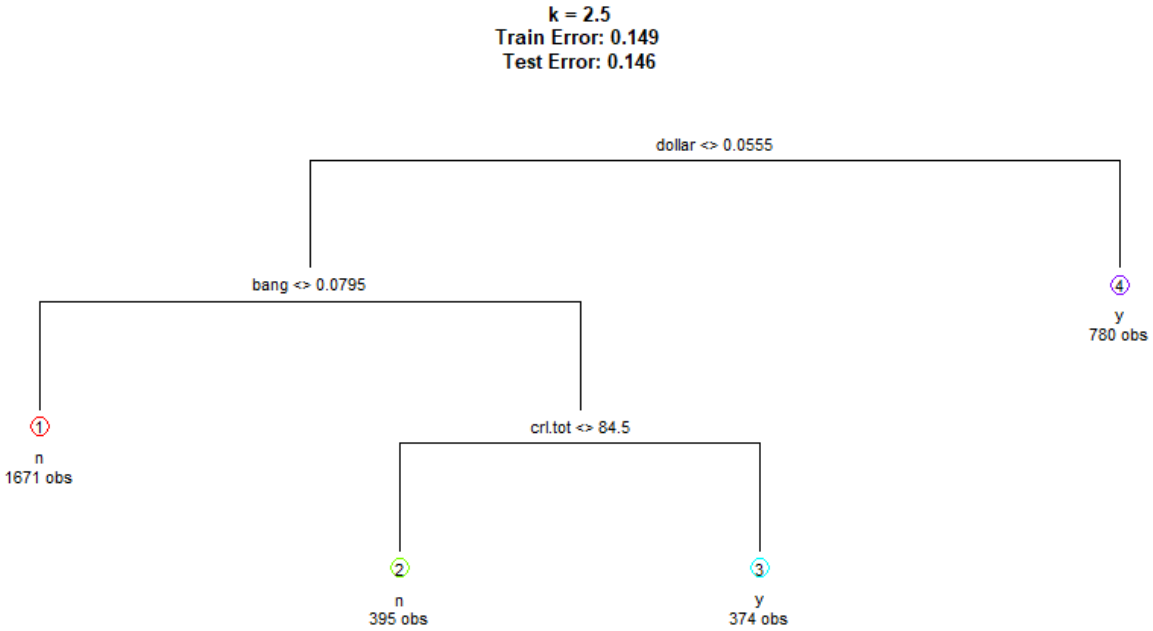

Рисунок 5 – Дерево решений k=2.5

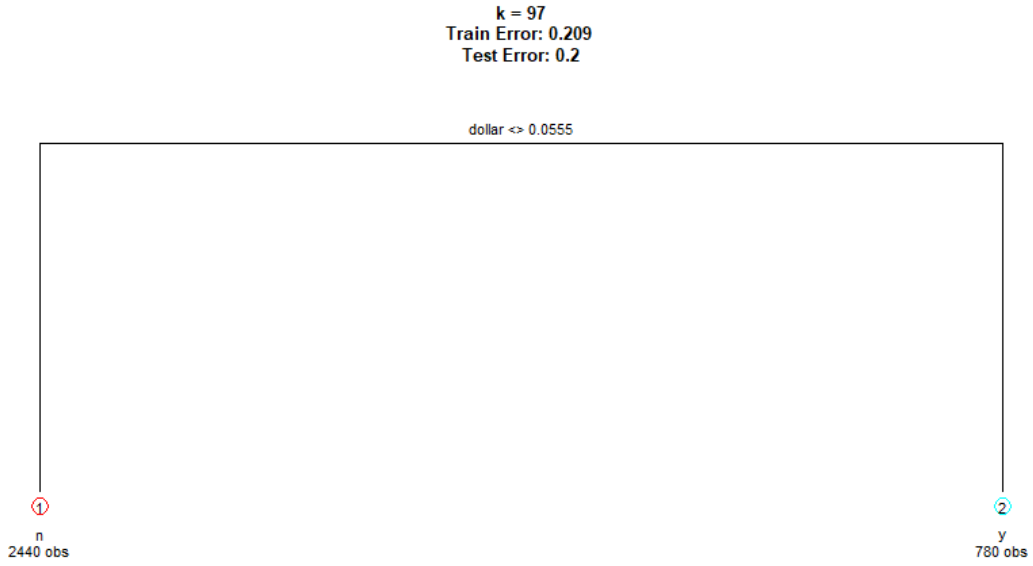

Рисунок 6 – Дерево решений k=97

Дерево с значением параметра k=0, является оптимальным, так как при увеличении этого параметра ошибка, как на тестовых, так и на тренировочных данных растет. Не смотря на большее количество узлов при k=0, дерево при нем не является избыточным.

Листинг кода 2 задачи:

```
library(DAAG)
data(spam7)
spam_data <- spam7

# Разделение данных на обучающую и тестовую выборки
set.seed(1235)
train_indices <- sample(1:nrow(spam_data), 0.7 * nrow(spam_data))
train_data <- spam_data[train_indices, ]
test_data <- spam_data[-train_indices, ]

# Построение дерева на обучающей выборке
spam_tree <- tree(yesno ~ ., data = train_data)
spam_tree_prune <- prune.misclass(spam_tree)

spam_tree_prune$k
#берем выведенные значения k (0.0, 2.5, 97.0) (-inf и 586 не учитываем)
k_values <- c(0.0, 2.5, 97.0)

for (k in k_values) {
  # Отсечение дерева
  pruned_tree <- prune.tree(spam_tree, k = k, method = 'misclass')

  # Предсказания
  train_predict <- predict(pruned_tree, train_data, type = "class")
  test_predict <- predict(pruned_tree, test_data, type = "class")

   # Оценка ошибки
  train_error <- sum(train_predict != train_data$yesno) / nrow(train_data)
  test_error <- sum(test_predict != test_data$yesno) / nrow(test_data)

  # График
  draw.tree(pruned_tree, cex = 0.7)
  title(paste(" k =", k, "\nTrain Error:", round(train_error, 3), "\nTest Error:", round(test_error,
3)),cex.main = 0.8)
}
```

Задание 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.

Решение

Построим регрессионное дерево (рисунок 7)

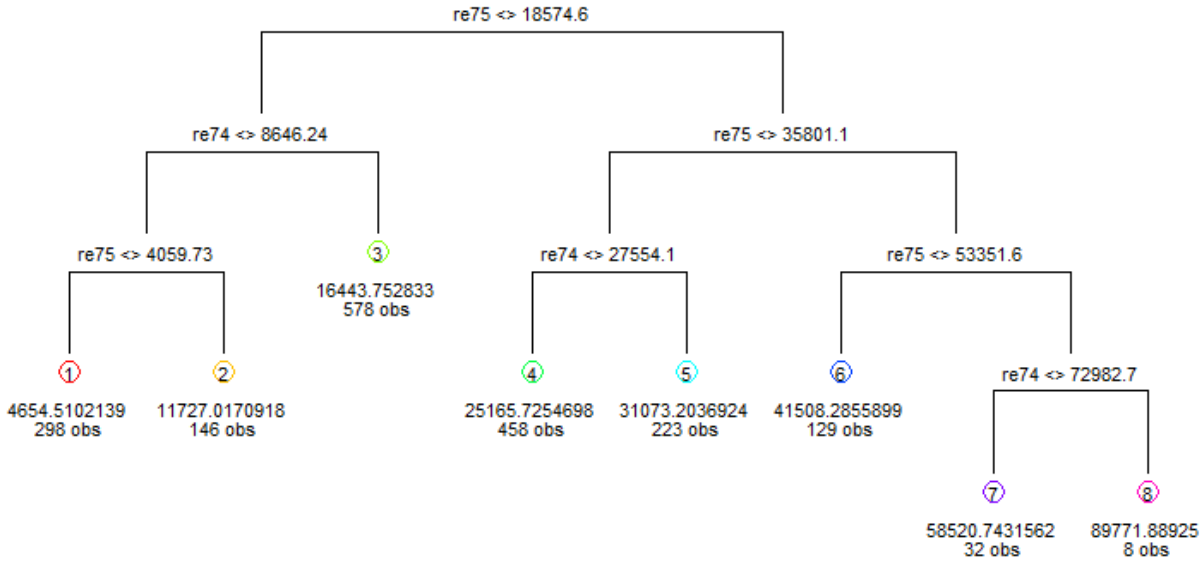

Рисунок 7 – Регрессионное дерево решений

Посчитаем среднеквадратические ошибки для модели регрессионного дерева и модели SVM-регрессии (рисунок 8)

```
Среднеквадратическая ошибка для регрессионного дерева: 107634908
> cat("Среднеквадратическая ошибка для svm модели:", mse_svm, "\n")
Среднеквадратическая ошибка для svm модели: 97653158
>
```

Рисунок 8 – MSE для регрессионного дерева и SVM-регрессии

Среднеквадратическая ошибка(отклонение) для регрессионного дерева равна 107634908. А для SVM-регрессии 97653158. Ошибка для SVM модели меньше, следственно данная модель более оптимальна для этих данных.

Листинг кода 3 задачи:

```
# Загрузка библиотек
library(DAAG)
library(e1071)

# Загрузка набора данных
data(nsw74psid1)
nsw_data<-nsw74psid1

# Разделение данных на обучающую и тестовую выборки
set.seed(1235)
train_indices <- sample(1:nrow(nsw_data), 0.7 * nrow(nsw_data))
train_data <- nsw74psid1[train_indices, ]
test_data <- nsw74psid1[-train_indices, ]

# регрессионное дерево
reg_tree <- tree(re78 ~ ., data = train_data)
draw.tree(reg_tree,cex=0.7)

# SVM-регрессия
svm_model <- svm(re78 ~ ., data = train_data,type = "eps-regression")

# Получение предсказаний на тестовой выборке
reg_tree_predict <- predict(reg_tree, newdata = test_data)
svm_predict <- predict(svm_model, newdata = test_data)

# Рассчет MSE для каждой модели
mse_tree <- mean((reg_tree_predict - test_data$re78)^2)
mse_svm <- mean((svm_predict - test_data$re78)^2)

# Вывод значений MSE
cat("Среднеквадратическая ошибка для регрессионного дерева:", mse_tree, "\n")
cat("Среднеквадратическая ошибка для svm модели:", mse_svm, "\n")
```

Задание 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?

Решение

Построим дерево решений (рисунок 9).

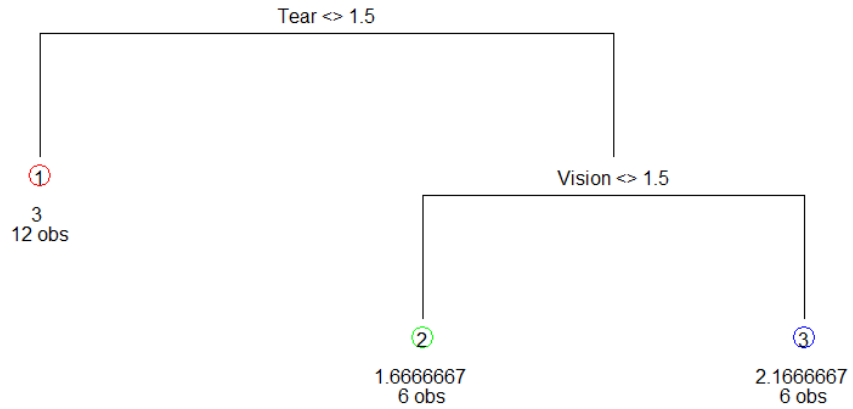

Рисунок 9 – Дерево решений

Предсказание при данном условие равно классу 3, то есть пациенту не следует носить контактные линзы

Листинг кода 4 задачи:

```
# чтение данных
lenses_data <- read.table("Lenses.txt",sep = "", stringsAsFactors = TRUE)
lenses_data <- lenses_data[,-1]

colnames(lenses_data) <- c("Age", "Vision", "Astigmatism", "Tear", "Class")

# Построение дерева решений
lenses_tree <- tree(Class ~ ., data = lenses_data)
draw.tree(lenses_tree)

# предсказание для условий
condition <- data.frame(Age = 2, Vision = 1, Astigmatism = 2, Tear = 1)
predict <- predict(lenses_tree, condition)

cat("Предсказанный тип:", predict, "\n")
```

Задание 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

Решение

Построим дерево решений с k=10 (prune) (рисунок 10).

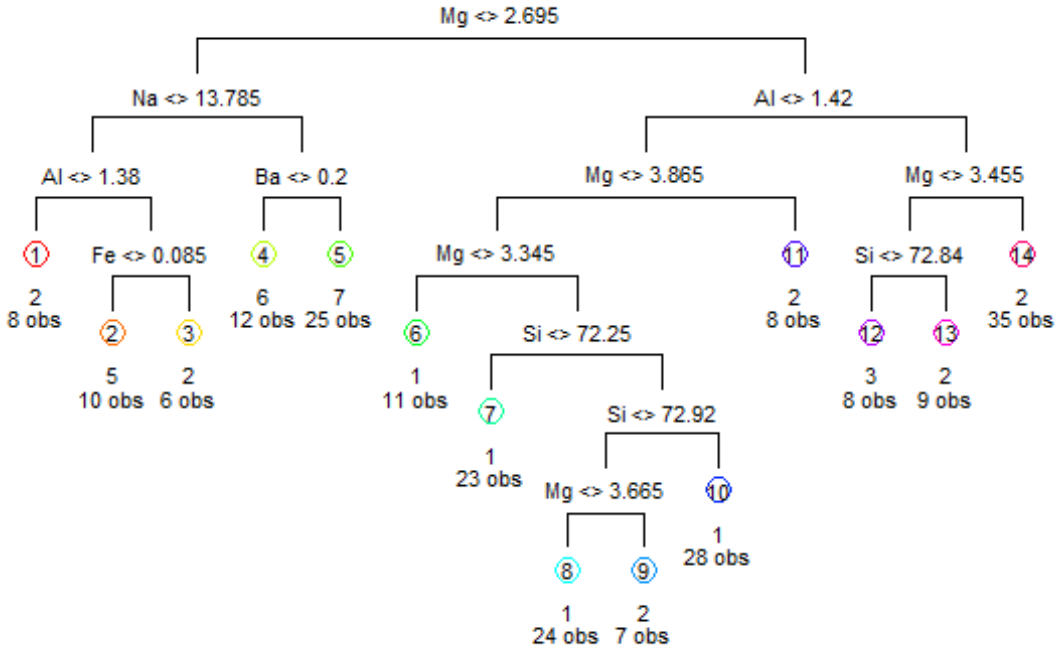

Рисунок 10 – Дерево решений

Экземпляр относится к типу стекла 2 **(**окна зданий, не плавильная обработка)

Листинг кода 5 задачи:

```
# Загрузка данных
data(Glass)
glass <- Glass
glass <- glass[, -1]

# Дерево решений
glass_tree5 <- tree(Type ~ ., data = glass)
glass_tree5_1 <- prune.tree(glass_tree5, k = 10)
draw.tree(glass_tree5_1,cex=0.7)

# Определение типа стекла
condition <- 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)
predict <- predict(glass_tree5_1, condition, type = "class")
cat("Glass Type:", predict, "\n")
```

Задание 6

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

Решение

Построим визуализацию тестовой выборки(рисунок 11).

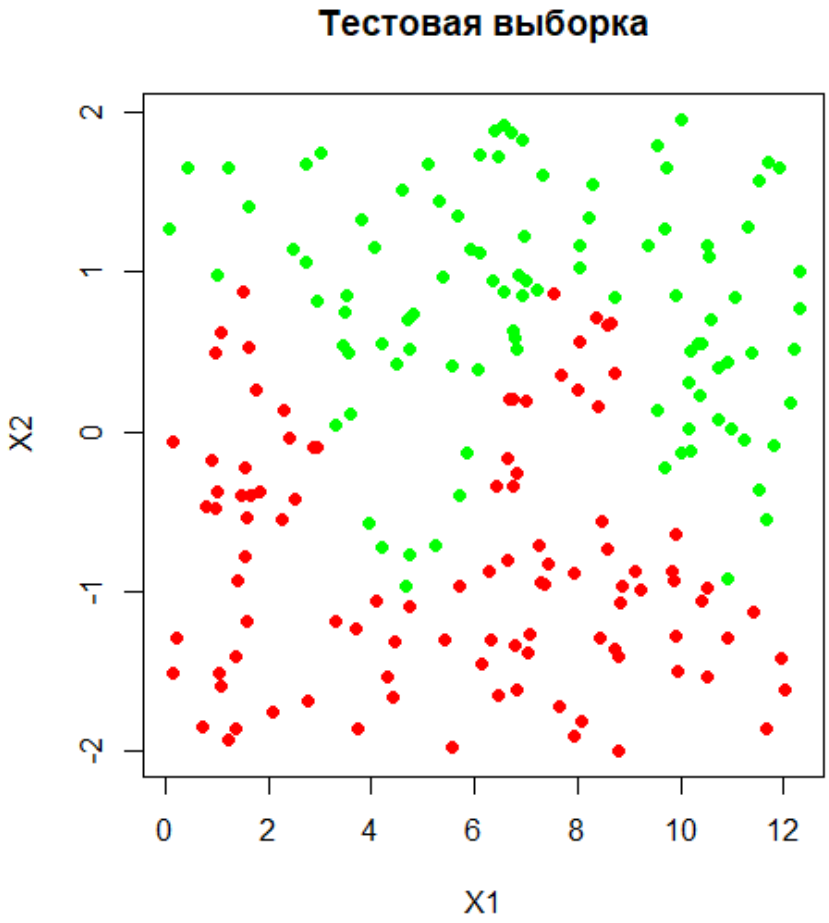

Рисунок 11 – Тестовая выборка

Построим график дерева решений без оптимизации. Визуализируем предсказание и посчитаем ошибку (рисунок 12).

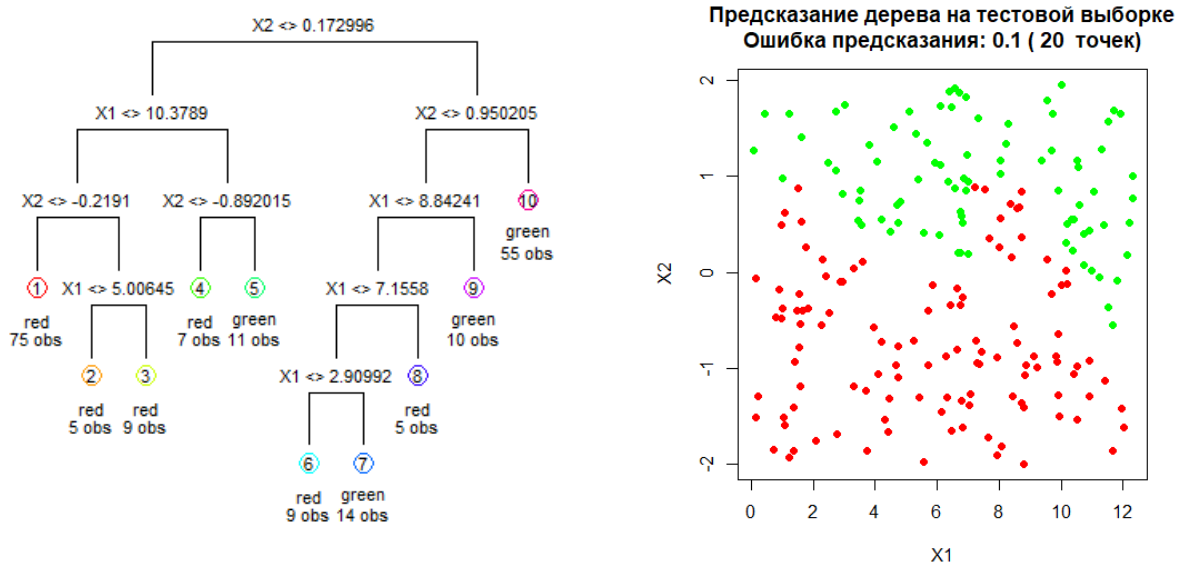

Рисунок 12 – Неоптимизированное дерево, результаты предсказаний

Ошибка равняется 0.1, 20 точек классифицировано неправильно. Оптимизируем дерево установив параметр k=11 (рисунок 13).

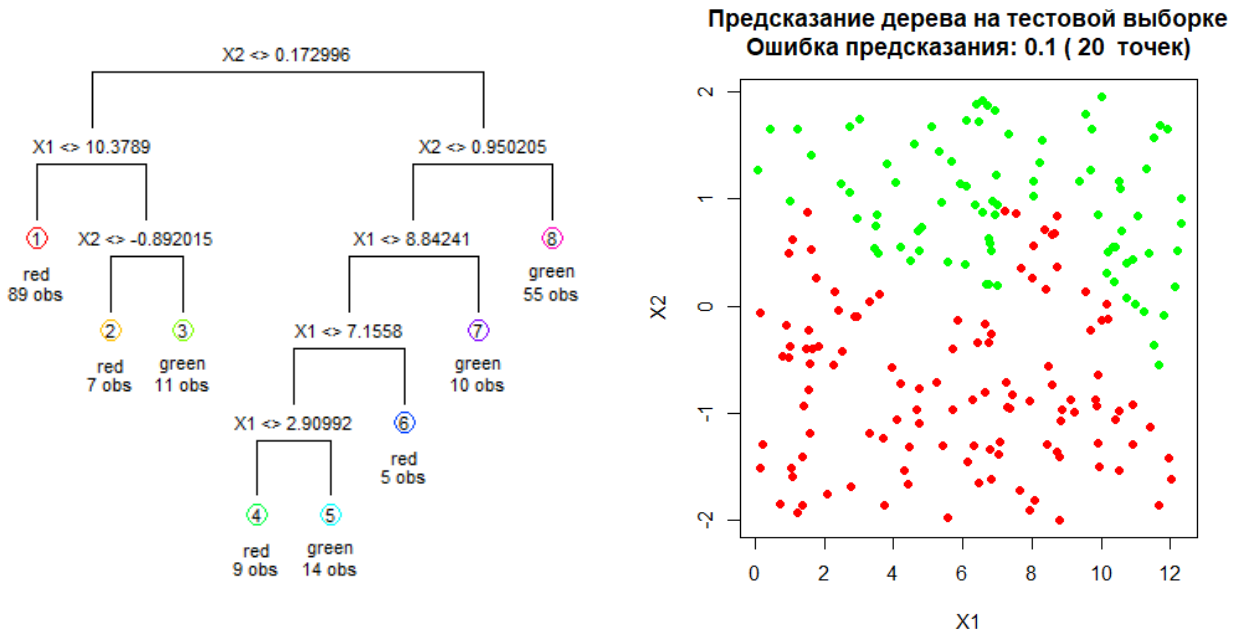

Рисунок 13 – Оптимизированное дерево k=11, результаты предсказаний

Количество узлов при этом уменьшилось, а ошибка осталось такой же. Тем самым мы избавились от избыточности дерева.

Листинг кода 6 задачи:

```
svm_data_tree <- function(train_data, test_data, k = NULL) {

  # Оптимизация дерева, если задан k параметр
  if (!is.null(k)) {
    tree_model <- prune.tree(tree_model, k = k)
  }

  draw.tree(tree_model, cex=0.7)

  # Предсказание
  test_predict <- predict(tree_model, newdata = test_data, type="class")

  # ошибка предсказания, пототочечная ошибка
  predict_error <- sum(test_predict != test_data$Colors) / nrow(test_data)
  point_error <- sum(test_predict != test_data$Colors)

  # График предсказания дерева на тестовой выборке
  par(pty = "s")
  plot(test_data$X1, test_data$X2, type = "n", xlab = "X1", ylab = "X2")
  points(test_data$X1, test_data$X2, col = class_colors[test_predict], pch = 19)
  title(paste("Предсказание дерева на тестовой выборке\nОшибка предсказания:",
              round(prediction_error, 3), "(", point_error, " точек)"))
}

# Загрузка данных
svm_data_train <- read.table("svmdata4.txt", sep = "\t", stringsAsFactors = TRUE)
svm_data_test <- read.table("svmdata4test.txt", sep = "\t", stringsAsFactors = TRUE)

# Цвета классов
class_colors <- c("green", "red")

# График тестовой выборки
par(pty = "s")
plot(svm_data_test$X1, svm_data_test$X2, type = "n", xlab = "X1", ylab = "X2")
points(svm_data_test$X1, svm_data_test$X2, col = class_colors[svm_data_test$Colors], pch =
19)
title("Тестовая выборка")

# Построение дерева решений
tree_model <- tree(Colors ~ ., data = svm_data_train)

# Дерево без оптимизации
svm_data_tree(svm_data_train, svm_data_test)

# Дерево с параметром k = 11
svm_data_tree(svm_data_train, svm_data_test, k = 11)
```

Задача 7.

Разработать классификатор на основе дерева решений для данных Титаник (Titanic dataset)

Решение

Построим дерево решений (рисунок 14)

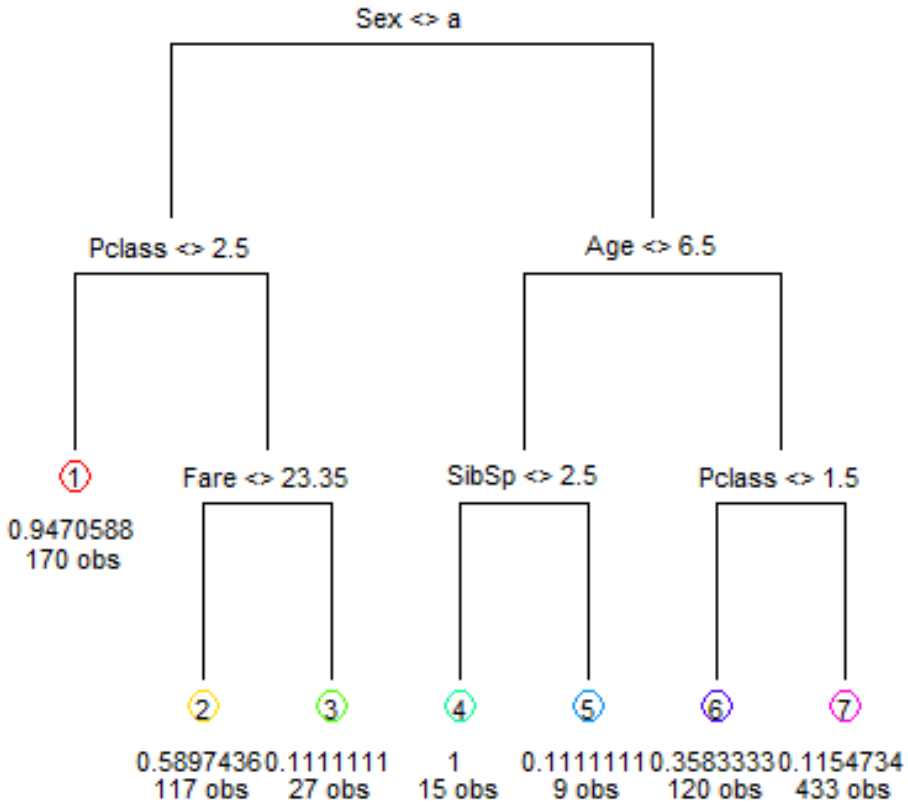

Рисунок 14 – Дерево решений

Получим результаты работы модели (рисунок 15)

```
Предсказанных выживших на тестовой выборке: 152
> cat("Предсказанных не выживших на тестовой выборке:", predicted_died, "\n")
Предсказанных не выживших на тестовой выборке: 266
> cat("точность на тренировочной выборке:", round(1-train_error, 3), "\n")
точность на тренировочной выборке: 0.827
>
```

Рисунок 15 – Результаты

Выживших (предсказание на тестовой выборке):152

Погибших (предсказание на тестовой выборке): 266

Точность модели на тренировочной выборке: 0.827

Листинг кода 7 задачи:

```
#функция подготовки данных
preprocess_data <- function(data) {
  # Удаляем столбцы "PassengerId", "Name", "Ticket", "Cabin"
  data <- data[, -which(names(data) %in% c("PassengerId", "Name", "Ticket", "Cabin"))]

  # Заполняем недостающие данные
  data$Age[is.na(data$Age)] <- mean(data$Age, na.rm = TRUE)
  data$Fare[is.na(data$Fare)] <- mean(data$Fare, na.rm = TRUE)
  embarked_mode <- names(sort(table(data$Embarked), decreasing = TRUE))[1]
  data$Embarked[is.na(data$Embarked)] <- embarked_mode

  # Кодируем категориальные признаки
  data$Sex <- as.factor(data$Sex)
  data$Embarked <- as.factor(data$Embarked)

  return(data)
}

# Загрузка данных
train_data_titanic <- read.csv("Titanic_train.csv")
test_data_titanic <- read.csv("Titanic_test.csv")

train_data_processed <- preprocess_data(train_data_titanic)
test_data_processed <- preprocess_data(test_data_titanic)

titanic_tree <- tree(Survived ~., train_data_processed)
draw.tree(model, cex=0.7)

# Предсказание на тестовой выборке
test_predict <- predict(titanic_tree,test_data_processed)

# Подсчет выживших и погибших
predicted_survived <- sum(round(test_predict))
predicted_died <- nrow(test_data_processed) - predicted_survived

# Оценка ошибки на тренировочной выборке
train_predict <- predict(titanic_tree,train_data_processed)
train_error <- sum(round(train_predict) != train_data_processed$Survived)/
nrow(train_data_processed)

cat("Предсказанных выживших на тестовой выборке:", predicted_survived, "\n")
cat("Предсказанных погибших на тестовой выборке:", predicted_died, "\n")
cat("точность на тренировочной выборке:", round(1-train_error, 3), "\n")
```