*Лабораторная работа №6*

*(часть 2)*

# Байесовская классификация и деревья принятия решений на R

## **6.5. Цель работы:**

Научиться выполнять классификацию на основе формулы Байеса и деревьев решений

## **6.6. Общие сведения**

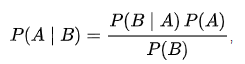
## 6.6.1 Наивный Байесовский подход

**Теорема Байеса** – одна из основных теорем теории вероятностей.

Теорема Байеса названа в честь её автора, английского математика и священника *Томаса Байеса* (англ.: *Thomas Bayes*) (1702 —1761).

Открытые ещё в 18-ом веке, формулы Байеса стали активно применяться с появлением компьютеров.

Формула условной вероятности наступления случайного события ***A*** в предположении, что имеет место событие ***В***, имеет вид:



(1)

где

P(A) — априорная вероятность гипотезы *A* (смысл такой терминологии см. ниже);

{\displaystyle P(A\mid B)} P(A|B)  — вероятность гипотезы *A* при наступлении события *B* (апостериорная вероятность);

{\displaystyle P(B\mid A)} P(B|A)  — вероятность наступления события *B* при истинности гипотезы *A*;

{\displaystyle P(B)} P(B)  — полная вероятность наступления события *B*

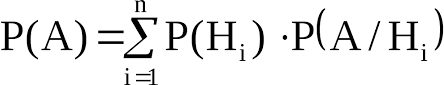
Формулу (1) можно переписать в виде:

𝑝(𝐴𝐵) =𝑝(𝐴|𝐵) ∙ 𝑝(𝐵) (2)

Предположим, что событие ***A*** наступает тогда и только тогда, когда происходит одно из ***n*** независимых и попарно несовместных событий, образующих *полную группу* (называемых *гипотезами* или *причинами*) *Hk*, т.е.

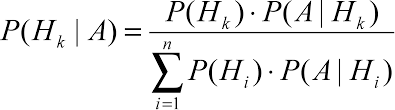
# A = A H1 + A H2 + … + A Hn. (3)

Тогда справедлива ***формула полной вероятности***:

(4)

Здесь p(Hi) – *априорные* вероятности гипотез, *i* = 1,…, *n*.

*Апостериорные* вероятности гипотез (т.е. вероятности гипотез в условиях наступления события ***A***) находятся (для каждого номера *k* = 1,…, *n*) по формулам Байеса:



## (5)

Заметим, что выражение в знаменателе этой формулы есть вероятность наступления события A — ***формула полной вероятности***.

Формулы Байеса, таким образом, позволяет «поменять местами» причину и следствие, т.е. по известному факту наступления события ***A*** вычислить вероятность того, что оно было вызвано данной причиной (*Hk*).

Одной из сфер применения формул Байеса является анализ данных, в частности, решение задач *классификации*, причём сырые данные (значения факторов), используемые для классификации, могут выражаться как *количественными*, так и *качественными* переменными.

## 6.6.2 Наивный Байесовский подход - пример

В среде R расчеты выполняются обычно с использованием функций NaiveBayes() из пакета klaR или naiveBayes() из пакета e1071.

Рассмотрим их использование на примере классификации цветков ириса:

library(klaR)

#Загрузка требуемого пакета: MASS

naive\_iris <- NaiveBayes(iris$Species ~ ., data = iris)

naive\_iris$tables

naive\_iris$tables$Petal.Width

# [,1] [,2]

# setosa 0.246 0.1053856

# versicolor 1.326 0.1977527

# virginica 2.026 0.2746501

Классификатор составил нам таблицы вероятностей по всем признакам, отобразить их нам поможет команда:

naive\_iris

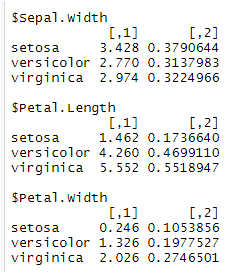


Рис 6.11. Средние значения параметров (первый столбец) и их стандартные отклонения (второй столбец) для каждого выделенного класса (фрагмент таблицы)

Для каждой метрической независимой переменной были выведены средние значения Petal.Width (первый столбец) и их стандартные отклонения (второй столбец) для каждого выделенного класса.

Можно установить, что ширина лепе стка у виргинского ириса существенно выше, чем у остальных двух видов. Для визуальной сравнительной оценки связи измеренных переменных с метками классов удобно рассмотреть ядерные функции плотности условной вероятности:

opar=par()

layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE))

plot(naive\_iris,lwd = 2, legendplot=TRUE)

#legend("topleft", legend=c("setosa", "versicolor", "virginica"),lty=1:3, cex=0.5)

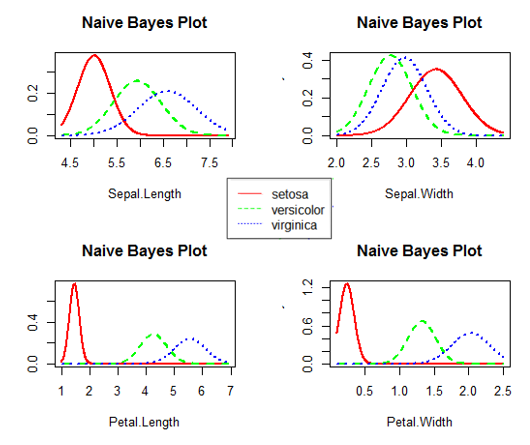


Рис 6.11. Ядерные функции плотности условной вероятности для таблицы native\_iris

par=opar #Восстанавливаем параметры графиков по умолчанию

Удалим пятый столбец таблицы, содержащий классификацию цветов и выполним

классификацию по "новым данным"

pred <- predict(naive\_iris, iris[, -5])$class

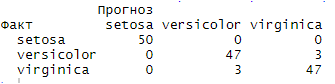
(table(Факт = iris$Species, Прогноз = pred)) 

Рис 6.12. Соотношение фактического состояния и прогноза для таблицы native\_iris

Оценим точность предсказаний для таблицы iris

Acc <- mean(pred == iris$Species)

Acc

#функция paste() - конвертирует векторы в текстовые переменные и объединяет их

#в одно текстовое выражение

paste("Точность=", round(100\*Acc, 2), "%", sep = "")

paste("Точность=", round(100\*Acc, 2), "%", sep = "")6.6. Деревья решений (пример)

## 

Рис 6.13. Оценка точности классификации ирисов согласно формуле Байеса

6.7. Классификация Decision Tree

*Подготовка данных*

> set.seed(1234)

> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))

> trainData <- iris[ind==1,]

> testData <- iris[ind==2,]

> nrow(trainData) # [1] 112

> nrow(testData) # [1] 38

> nrow(iris) # [1] 150

На данном шаге мы разбиваем исходные данные на две выборки обучающую и тестовую. Для этого сначала используем функцию sample(), которая вернет нам вектор значений 1 или 2, размером, равным количеству строк в исходной выборке ( за это отвечает функция nrow() ), вероятность выпадения 1 равна 70%, а 2 – 30%. Далее, используя логическое индексирование. разобьем исходные данные.

*Построение модели*

> install.packages("party")

> library(party)

> myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width

> iris\_ctree <- ctree(myFormula, data=trainData)

**Для построения модели сначала необходимо задать какой параметр (целевой признак) мы хотим предсказать и от каких признаков он зависит.**

Для этого используется оператор ~ ( тильда ). Если мы хотим добавлять признаки по одному, то используется +, если хотим показать зависимость от всех признаков сразу, то используем оператор ~. , если же нужно исключить некоторые признаки, то используем -. Например, **Species ~. – Sepal.Length.** Запись показывает, что следует предсказывать целевой признак Species по всем признакам кроме Sepal.Length.

Следующим шагом строим модель, используя функцию ***ctree()***, передав ей в качестве параметра нашу формулу и выборку для обучения модели (в нашем случае это trainData ).

*Обучение модели*

> table(predict(iris\_ctree), trainData$Species)

setosa versicolor virginica

setosa 40 0 0

versicolor 0 37 3

virginica 0 1 31

Для применения модели используется функция **predict().** По умолчанию функция **predict()** применяется к данным, которые использовались для построения/обучения модели. Далее строится так называемая ***Confusion Matrix*** (матрица ошибок или путаницы, если переводить дословно).

Читать данную таблицу следует так, например, при предсказанном значении versicolor, реально 3 объекта имели класс virginica.

*Визуализация*

>plot(iris\_ctree)

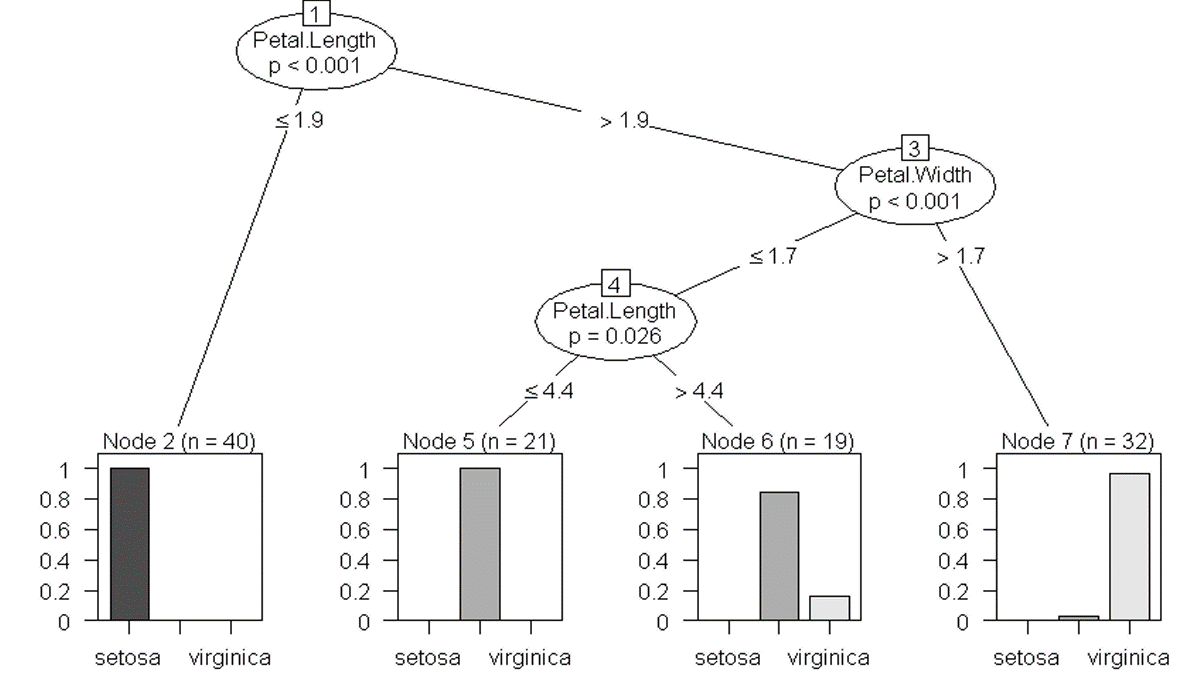


Рис 6.14. Классификация ирисов с помощью двоичного дерева (стадия обучения).

Чтобы применить ***обученную модель*** для произвольной выборки, необходимо использовать функцию **predict** с параметром **newdata**.

*Применение модели*

>test\_predicted <- predict(iris\_ctree, newdata=testData)

> table(test\_predicted, testData$Species)

setosa versicolor virginica

setosa 10 0 0

versicolor 0 12 2

virginica 0 0 14

|  |
| --- |
|  |
| Рис 6.15. Классификация ирисов с помощью двоичного дерева. |

Видно, что классификация сработала хорошо, мы получили два ошибочных элемента при классификации вида versicolor, что составляет 1,7%.

***6.8. Алгоритм Random Forest***

Random Forest - один из наиболее популярных алгоритмов машинного обучения или data mining.

Во-первых, он невероятно универсален, с его помощью можно решать как задачи регрессии так и классификации. Проводить поиск аномалий и отбор предикторов.

Во-вторых, это тот алгоритм, который действительно сложно применить неправильно. Просто потому, что в отличии от других алгоритмов у него мало настраиваемых параметров. И еще он удивительно прост по своей сути. И в то же время он отличается высокой точностью.

Главная идея алгоритма - обучение ансамбля в действии. Алгоритм Random Forest потому и называется "Случайный Лес", что для полученных данных он создает множество деревьев приятия решений и потом усредняет результат их предсказаний. Важным моментом тут является элемент случайности в создании каждого дерева - если мы создадим много одинаковых деревьев с разной точностью, то результат их усреднения будет обладать точностью одного дерева.

*Построение модели*

> library(randomForest)

> rf <- randomForest(Species ~ .,data=trainData, ntree=100, proximity=TRUE)

> table(predict(rf), trainData$Species)

setosa versicolor virginica

setosa 40 0 0

versicolor 0 35/37 3/3

virginica 0 2/1 32/31

Здесь числа, помеченные красным, написаны для сравнения с алгоритмом дерева решения.

Работа с алгоритмом randomForest абсолютно аналогична применению алгоритма Decision Tree. Для работы с алгоритмом необходимо подключить пакет rendomForest и использовать одноименную функцию.

В Вашей задаче, возможно, придется уменьшить количество деревьев (ntree) для получения адекватного результата.

*Информация о модели*

>print(rf)

Call: randomForest(formula = Species ~ ., data = trainData, ntree = 100, proximity = TRUE)

Type of random forest:classification

Number of trees: 100

No. of variables tried at each split: 2

OOB estimate of error rate: 4.46%

Confusion matrix: #матрица ошибок – показывает долю отклонений

setosa 40 0 0 0.00000000

versicolor 0 35 3 0.07894737

virginica 0 2 32 0.05882353

*Альтернативная реализация*

>library(party)

>cf <- cforest(Species ~ ., data=trainData, control=cforest\_unbiased(mtry=2,ntree=100))

> table(predict(cf), trainData$Species)

## **6.8. Формулировка задания:**

## **Данные берутся из предыдущего набора (ЛР 6 *часть 1*). Вам необходимо добавить найденные при в результате кластерного анализа классы (groups), как вектор-столбец в предыдущий DATASET, разделить его на две части (обучающую и тестовую) обучить на обучающей выборке классификатор, а затем применить классификатор к тестовым данным. Важно: понадобится преобразовать вектор groups в фактор.**

## Для наивного Байесовского классификатора:

1. Решите задачу с помощью наивного Байесовского классификатора;

2. Проанализируйте точность полученных решений для тестовых данных

## Для деревьев решений:

1. Примените метод деревьев решений для задачи классификации (для того же набора данных).

2. Исследуйте дерево решений; если позволяет размерность, постройте его график.

3. Проанализируйте точность полученных решений для тестовых данных (с известным значением переменной отклика), сравните результаты с ранее полученными.

4. Выполнить классификацию с помощью случайного леса, сопоставить результат с результатом дерева решения, прокомментировать результат сравнения.

5. Сопоставьте результаты с результатами Байесовского классификатора.

6. Проанализируйте полученные результаты, сделайте вывод относительно результатов кластерного анализа (ЛР 6.1), оформите общий отчёт по ЛР 6.1 и 6.2.

Для выполнения заданий использовать данные из части 6.1.