МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Методичні вказівки
до виконання лабораторних робіт 1-9
для добувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 «Комп’ютерні науки»
та F6 «Інформаційні системи та технології

Київ 2025

УДК 004.6
І-73

Укладачі: С. В. Білощицька, д-р техн. наук, доцент,
І. В. Босенко, д-р філософії,
О. О. Маційовський, асистент

Рецензент І. А. Ачкасов, д-р техн. наук, професор

Відповідальна за випуск Т. А. Гончаренко, д-р техн. наук, професор

Затверджено на засіданні кафедри інформаційних технологій, протокол № 4 від 27 жовтня 2025 року.

В авторській редакції.

Інтелектуальний аналіз даних [електронний ресурс]:
І-73 методичні вказівки до виконання лабораторних робіт 1-9 / уклад.: Білощицька С.В., Босенко І.В., Маційовський О.О. – Київ : КНУБА, 2025. – 66 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначено для здобувачів першого (бакалаврського))рівня вищої освіти спеціальностей F3 «Комп’ютерні науки» та F6 «Інформаційні системи та технології».

Зміст

Загальні положення

Методичні вказівки до дисципліни «Інтелектуальний аналіз даних» мають на меті забезпечити якісну підготовку здобувачів до виконання лабораторних робіт, що є невід'ємною складовою практичного опанування методів статистичного аналізу, машинного навчання та обробки даних. Успішне виконання лабораторних робіт ґрунтується на засвоєнні теоретичного матеріалу, поданого на лекціях, а також на здатності застосовувати відповідні математичні та програмні інструменти під час роботи з реальними даними.

Здобувачам рекомендується особливу увагу приділити базовим положенням, що лежать в основі лабораторних завдань, зокрема принципам роботи з мовою R, правилам завантаження та опрацювання даних, методам побудови моделей, перевірки статистичних гіпотез, кластеризації та візуалізації результатів. У методичних вказівках наведено приклади використання функцій та алгоритмів, які слугують орієнтиром, але не обмежують здобувача у виборі альтернативних коректних способів розв'язання поставлених задач.

У процесі виконання лабораторних робіт здобувач має продемонструвати здатність до самостійного аналізу даних, правильного застосування статистичних методів, інтерпретації отриманих результатів та оцінювання якості побудованих моделей. Важливим є також вміння структурувати дані, налагоджувати код, проводити візуальний аналіз графіків, а також обґрунтовувати вибір тих чи інших алгоритмів. Не менш значущим є вміння документувати виконані дії, структурувати результати та робити висновки на основі проведених експериментів.

Перед виконанням лабораторної роботи здобувач повинен опрацювати методичні вказівки, теоретичні відомості, приклади коду та пояснення алгоритмів, а також ознайомитися з вимогами до звіту. Під час практичної реалізації завдання необхідно ретельно перевіряти правильність введення даних, коректність виконання функцій та уникати помилок.

Методичні вказівки є ключовим інструментом для системного та послідовного опанування дисципліни «Інтелектуальний аналіз даних», оскільки вони забезпечують практичне засвоєння сучасних методів аналізу інформації, сприяють розвитку професійних компетенцій здобувачів та формують навички застосування інтелектуальних технологій у майбутній професійній діяльності.

ЛАБОРАТОРНА РОБОТА № 1

Тема: Основи роботи в системі R

Мета: Навчитися працювати в системі R

Теоретичні відомості

R – це мова програмування та середовище для статистичних обчислень і графіки, яка широко використовується в інтелектуальному аналізі даних, машинному навчанні та статистиці. Основні можливості R включають:

- Генерацію випадкових даних (наприклад, нормальних, рівномірних).
- Обробку та аналіз даних за допомогою фреймів даних.
- Побудову графіків для візуалізації результатів.
- Реалізацію методів моделювання, таких як метод найменших квадратів.

RStudio – це інтегроване середовище розробки для R, яке робить роботу з R зручною завдяки графічному інтерфейсу та інструментам для налагодження, візуалізації та організації проєктів.

Для генерації випадкових чисел в R використовуються спеціальні функції:

rnorm(N, mean, sd) — генерує ***N*** випадкових чисел із нормального розподілу з математичним сподіванням mean та стандартним відхиленням sd.

runif(N,* min, max*) — генерує ***N*** випадкових чисел із рівномірного розподілу на відрізку ***[*min, max*]***.

В нормальному розподілі математичне сподівання (μ) є середнім значенням вибірки, а стандартне відхилення (σ) показує, наскільки дані розкидані навколо середнього.

Квантиль розділяє дані таким чином, що певна частина значень вибірки розташована нижче цього значення, а решта — вище.

P-квантиль (q_r) — це значення, для якого частка ***r*** вибірки є меншою або рівною цьому значенню.

У R функція ***quantile()*** використовується для обчислення квантилів.

R підтримує різні математичні функції, такі як:

log() — натуральний логарифм.

log2() — логарифм за основою 2. Ці функції використовуються для моделювання залежностей, таких як розрахунок заробітної плати в одному із наступних завдань.

Завдання

1. Згенерувати вектор довжини N=1000, елементами якого є нормально розподілені випадкові величини з математичним сподівання 1 і стандартним квадратичним відхиленням 0,3. Порахувати математичне сподівання та стандартну помилку не використовуючи вбудовані функції. Перевірити правильність розрахунків. Порахувати .95,.99-квантилі. Дослідити відхилення статистичного математичного сподівання при зростанні обсягу вибірки N (N=1000,2000,4000,8000).
2. Створити фрейм даних з N=20 записів з полями: Nrow – номер запису, Name – ім'я користувача, BirthYear – рік народження користувача, EmployYear – рік прийому на роботу, , Salary – заробітна плата, де Nrow змінюється от 1 до N, Name задається довільно, BithYear розподілено рівномірно на відрізку [1960,1985], EmployYear розподілено рівномірно на відрізку [BirthYear+18,2006], Salary для співробітників молодше 1975 р. н. визначається за формулою Salary=(ln(2007-EmployYear)+1)*8000, для інших Salary=(log2(2007-EmployYear)+1)*8000. Порахувати число співробітників із заробітною платою, більшою 15000. Додати в таблицю поле, що відповідає сумарному податку з прибутку (ставка 13%)сплаченою співробітником за час його роботи в організації, якщо його заробітна плата за кожний рік роботи розраховувався за наведеними вище формулами.
3. Спроєктувати та реалізувати метод найменших квадратів.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв'язання задач.
5. Висновки.

Контрольні запитання

1. Яка вбудована функція системи R використовується для генерації нормально розподілених чисел?
2. Яка вбудована функція системи R використовується для генерації рівномірно розподілених чисел?
3. Поясніть суть методу найменших квадратів.

ЛАБОРАТОРНА РОБОТА № 2

Тема: Перевірка статичних гіпотез

Мета: Навчитися перевіряти статистичні гіпотези до допомогою вбудованих функцій системи R

Теоретичні відомості

Інформація, яку дістають на підставі вибірки, реалізованої із генеральної сукупності, може бути використана для формулювання певних суджень про всю генеральну сукупність. Наприклад, розпочавши виготовляти кондитерські вироби певного типу, відбирають певну кількість цих виробів і піддають її певним тестам.

За результатами тестів можна зробити висновок про те, чи кращі нові вироби від виробів старого типу, чи ні. А це, у свою чергу, дає підставу для прийняття рішень: виготовляти їх чи ні.

Такі рішення називаються *статистичними*. Статистичні рішення мають імовірнісний характер, тобто завжди існує ймовірність того, що прийняті рішення будуть помилковими. Головна цінність прийняття статистичних рішень полягає в тому, що в мережах імовірнісних категорій можна об’єктивно виміряти ступінь ризику, що відповідає тому чи іншому рішенню.

Будь-які статистичні висновки, здобуті на підставі обробки вибірки, називаються *статистичними гіпотезами*.

Статистичні гіпотези про значення параметрів ознак генеральної сукупності називаються *параметричними*. Наприклад, висувається статистична гіпотеза про числові значення генеральної середньої $\bar{X}_r$, генеральної дисперсії D_r , генерального середнього квадратичного відхилення σ_r та ін.

Статистичні гіпотези, що висуваються на підставі обробки вибірки про закон розподілу ознаки генеральної сукупності, називаються *непараметричними*. Так, наприклад, на підставі обробки вибірки має бути висунута гіпотеза, що ознака генеральної сукупності має нормальний закон розподілу, експоненційний закон та ін.

Гіпотезу, що підлягає перевірці, називають *основною*. Оскільки ця гіпотеза припускає відсутність систематичних розбіжностей (нульові розбіжності) між невідомим параметром генеральної сукупності і величиною, що одержана внаслідок обробки вибірки, то її називають

нульовою гіпотезою і позначають → H_0. *Вбудована в системі R нульова гіпотеза стверджує, що випадкова величина x розподілена за нормальним законом.* Для підтвердження або заперечення того, що вибірка випадкової величини розподілена за нормальним законом, існує вбудована функція ***shapiro.test(x)***, в якій реалізовано тест Шапіро-Уілка. При цьому розмір вибірки повинен становити не менше 3 та не більше 5000 елементів.

Функція: ***shapiro.test(x)*** повертає список, який включає поля:

statistics – значення статистики Шапіро–Уілка;

p.value – рівень значущості;

method – назва тесту «***Shapiro-Wilk normality test***»;

data.name – назва даних, які тестувалися.

Приклад 1. Фрейм даних ***trees*** з бібліотеки ***datasets*** містить інформацію про діаметр, висоту та об'єм вишневих дерев. Перевірити гіпотезу про те, що висота дерев розподілена за нормальним законом при рівні значимості α=0.05.

Розв'язання:

```
> colnames(trees)
[1] "Girth"  "Height" "Volume"
> trees$Height
        [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64 78 80 74 72
77 81 82 80 80 80 87
> x=trees [, "Height"]
> hist(x,col="green", xlab="Tree heights", main="Tree height frequencies")
> shapiro.test (trees$Height)
  Shapiro-
Wilk
normality test
data:
```

При рівні значущості α=0.05 гіпотеза про нормальний закон розподілу висот вишневих дерев приймається, оскільки p - *value* >α. Візуалізацію результату у вигляді гістограми можна побачити на рис. 1.

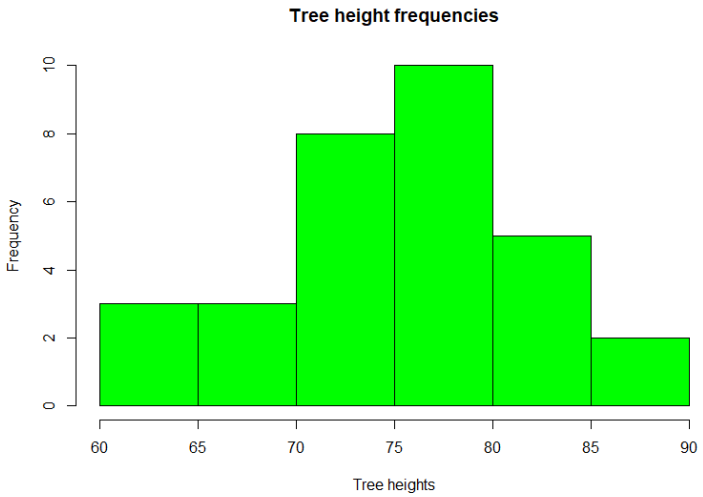

Рис. Гістограма частот висоти вишневих дерев

Критерій узгодженості Колмогорова-Смирнова використовується для того, щоб визначити, чи підпорядковуються два емпіричних розподіли одному закону, або визначити, чи підпорядковується емпіричний розподіл певній моделі.

Вбудована в системі R функція критерію Колмогорова-Смирнова

ks.test(x, y, ..., alternative = c("two.sided", "less", "greater"), exact = NULL),

де

x – вектор, що містить першу вибірку;

y – вектор, що містить другу вибірку;

... – параметри розподілу;

alternative – одне зі значень, що позначає тип альтернативної гіпотези:

«***two.sided***» (за промовчуванням) означає, що інтегральна функція розподілу вибірки ***x*** не відповідає гіпотетичній,

«***less***» означає, що інтегральна функція розподілу вибірки ***x*** не більше гіпотетичної,

«***greater***» означає, що інтегральна функція розподілу вибірки ***x*** не менша гіпотетичної;

повертає список, який містить поля:

statistics – значення статистики Колмогорова-Смирнова;

p.value – рівень значущості;

alternative – символьний рядок в якому наводиться опис альтернативної гіпотези;

method – назва тесту «Kolmogorov-Smirnov test»;

data.name – назва даних, які тестувалися.

Приклад 2. Фрейм даних ***randu*** з бібліотеки ***datasets*** містить три вибірки псевдо-випадкових чисел з інтервалу [0,1]. Розмір кожної вибірки – 400 елементів. Послідовності представлені у вигляді матриці з трьома стовпцями, які відповідно мають назви x, y, z.

Розв'язання:

```
> colnames(randu)
      [1] "x" "y" "z"
> ks.test (randu$x,randu$y)
  Two-sample Kolmogorov-
Smirnov test data: randu$x and
randu$y
      D = 0.085, p-value = 0.1111
      alternative hypothesis: two-sided
> ks.test (randu$x,randu$z)
> ks.test (randu$y,randu$z)
> ks.test (randu$x,punif)
> ks.test (randu$y,punif)
> ks.test (randu$z,punif)
```

t-тест Ст'юдента – це загальна назва класу методів перевірки статистичних гіпотез, які базуються на порівнянні з розподілом Ст'юдента.

Система R надає можливість проводити t-тест Ст'юдента на одній або двох вибірках. Тест Ст'юдента для *однієї вибірки* призначений для перевірки гіпотези про те, що середнє значення нормально розподіленої генеральної сукупності дорівнює деякому заданому значенню, а дисперсія не відома.

Тест Ст'юдента для *двох вибірок* призначений для перевірки гіпотези, про те, що середні значення двох нормально розподілених генеральних сукупностей є однаковими.

Вбудована в системі R функція перевірки гіпотези за допомогою t-тесту Ст'юдента для однієї вибірки має синтаксис:

t.test(x, ...)

Вбудована в системі R функція для перевірки гіпотези за допомогою t-тесту Ст'юдента для двох вибірок має синтаксис:

t.test(x, y = NULL,

alternative = c("two.sided", "less", "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)

t.test(formula, data, subset, na.action, ...)

де

x – вектор, що містить першу вибірку;

y – вектор, що містить другу вибірку;

... – параметри розподілу;

alternative – одне зі значень, що позначає тип альтернативної гіпотези:

«***two.sided***» – (за промовчуванням) означає, що інтегральна функція розподілу вибірки *x* не відповідає гіпотетичній,

«***less***» означає, що інтегральна функція розподілу вибірки ***x*** не більше гіпотетичної,

«***greater***» означає, що інтегральна функція розподілу вибірки ***x*** не менша гіпотетичної;

mu – математичне сподівання або різниця математичних сподівань, якщо задано дві вибірки;

paired – логічне значення, що вказує, чи потрібно виконувати спільний t-тест;

var.equal – логічне значення, яке визначає, чи потрібно вважати розкиди однаковими. Якщо TRUE, то обчислюється розкид для спільної вибірки. ***conf.level*** – ступінь довіри;

formula – формула вигляду ***lhs~rhs***, де ***lhs*** – числовий вектор, ***rhs*** – фактор з двома класами. Має місце тільки для тесту з двома вибірками.

data – масив або фрейм з якого зчитуються дані;

subset – вектор, який вказує на використовувану підмножину спостережень;

na.action – функція, ініціалізація якої відбувається тоді, коли в даних з'являється значення ***NA***;

повертає список, який містить поля:

statistics – значення t-статистики Ст'юдента;

parameter - число ступенів свободи;

p.value – рівень значущості;

conf.int – довірчий інтервал для математичного сподівання;

estimate – оцінка математичного сподівання при тестуванні однієї вибірки або різниця математичного сподівання при тестуванні двох вибірок;

alternative – символьний рядок в якому наводиться опис альтернативної гіпотези;

method – символьний рядок в якому зазначається назва тесту;

data.name – назва даних, які тестувалися.

Приклад 3. Порівняємо математичне сподівання вибірок, згенерованих за допомогою функції ***rnorm***.

Розв'язання:

```
>     set.seed(0)
>     x=rnorm(100,mean=0,sd=4)
>     y=rnorm(100,mean=1,sd=4)
>     t.test(x,y)

        Welch Two Sample

t-test data:  x and y
        t = -1.3896, df = 196.428, p-value = 0.1662
        alternative hypothesis: true difference in means is not
equal to 0 95 percent confidence interval:
        -1.7590435  0.3048036
        Sample
estimates:  mean  of  x
mean  of  y  0.0906738
```

Знайдене значення t-статистики становить -1,3896, число ступенів свободи – 196, 428, розрахований рівень значущості гіпотези – 0,1662. Вказані граничні 95% довірчого інтервалу для різниці математичного сподівання першого та другого розподілів. При рівні значущості $\alpha=0.1$ гіпотеза про те, що математичне сподівання розподілів є однаковим, приймається, оскільки p - value > α.

Перевіримо альтернативну гіпотезу, яка полягає в тому, що математичне сподівання другої вибіркової сукупності є більшим ніж першої вибіркової сукупності.

```
> t.test(x,y,alternative="less")

     Welch  Two  Sample  t-

test data:  x and y
        t = -1.3896, df = 196.428, p-value = 0.08311
        alternative hypothesis: true difference in means is
less than 0 95 percent confidence interval:
            -Inf 0.1376404
        sample
estimates:  mean  of  x
mean of y
```

Тепер при рівні значущості α=0.1 нуль-гіпотеза відхиляється і приймається альтернативна гіпотеза, оскільки p - value <α.

Критерій узгодженості Пірсона χ^2 призначений для перевірки гіпотези про незалежність ознак сукупності.

Вбудована в системі R функція використання критерію узгодженості Пірсона має синтаксис:

chisq.test(x, y = NULL, correct = TRUE,

p = rep(1/length(x), length(x)), rescale.p = FALSE, simulate.p.value = FALSE, B = 2000)

де

x – вектор або матриця;

y – вектор. Працює у тих випадках, коли x не є матрицею;

correct – логічне значення, що вказує, чи потрібно застосовувати неперервне корегування для матриць розміром 2×2;

p – вектор ймовірностей. Повинен мати таку ж довжину, що й ***x*** ;

rescale.p – логічне значення. Якщо ***TRUE***, то при p за потреби масштабується, таким чином, що сума його компонентів дорівнює 1;

simulate.p.value – логічне значення. Якщо ***TRUE***, то ***p.value*** обчислюється

методом Монте-Карло, в протилежному випадку використовується χ^2 розподіл;

B – кількість випробувань за методом Монте-Карло.

повертає список, який містить поля:

statistics – значення χ^2 -статистики Пірсона;

parameter – число ступенів свободи розподілу χ^2 , може дорівнювати ***NA***, якщо для обчислення ***p.value*** використовується метод Монте-Карло;

p.value – рівень значущості;

method – символьний рядок в якому зазначається назва тесту, а також назва методу корегування: неперервний чи метод Монте-Карло;

data.name – назва даних, які тестувалися;

observed – число точок, які потрапили в і -й інтервал групування (дорівнює x на вході, якщо x – вектор, а y не використовується);

expected – теоретичне число точок (в припущенні про виконання гіпотези), які потрапили в і -й інтервал групування;

residuals – залишок Пірсона.

Приклад 4. Перевіримо гіпотезу про незалежність двох випадкових величин. В таблиці ***HairEyeColor*** з бібліотеки *datasets* наведені стать, колір волосся та колір очей 592 студентів. Таблиця має такі стовпці: ***"Hair": HairEyeColor["Black", ,], HairEyeColor["Brown", ,], HairEyeColor["Red", ,], HairEyeColor["Blond", ,],"Eye": HairEyeColor[, "Brown",], HairEyeColor[, "Blue",], HairEyeColor[, "Hazel",], HairEyeColor[, "Green",],"Sex": HairEyeColor[, , "Male"], HairEyeColor[, , "Female"].***

Елементи таблиці – кількість осіб даної групи. Перевіримо нуль-гіпотезу про те, що колір очей чоловіків не залежить від кольору їх очей. Для цього виконаємо наступні дії.

```
> men=HairEyeColor[,,'Male']
> men
       Eye
Hair    Brown Blue Hazel Green
 Black   32  11    10    3
 Brown   53  50    25    15
 Red     10  10    7     7
 Blond   3   30    5     8
> mosaicplot(men,col=c('chocolate','cornflowerblue','salmon','green'),main='Male eye color vs.hair color')
> chisq.test(men,simulate.p.value=TRUE)

        Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)

data:  men
X-squared = 41.2803, df = NA, p-value = 0.0004998
```

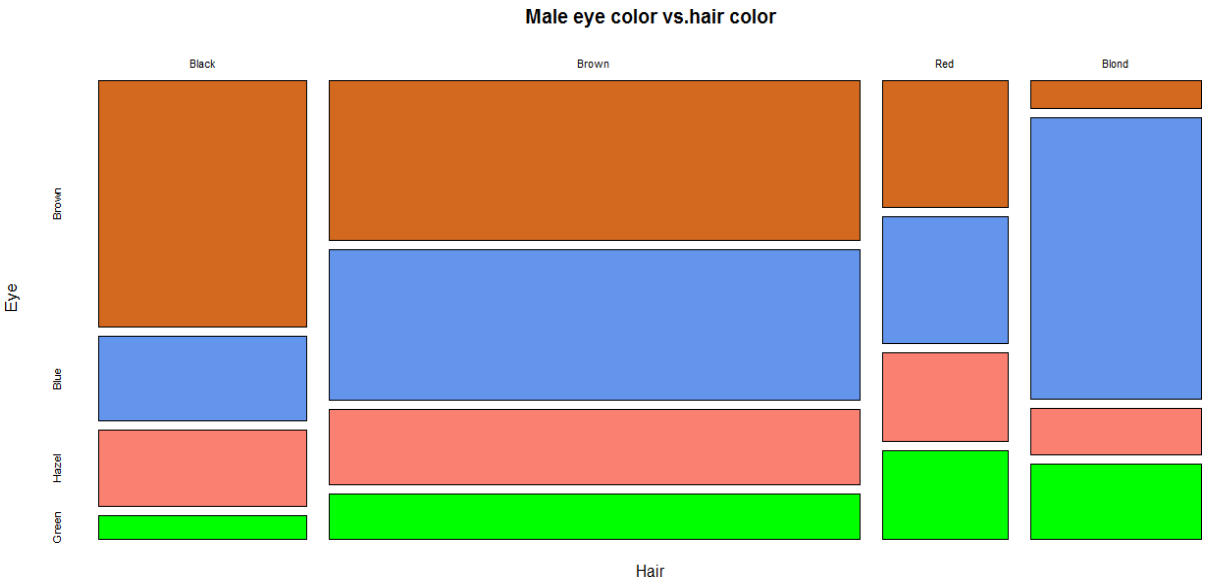

Рис. Залежність кольору очей чоловік від кольору їх очей

При рівні значущості α=0.05 гіпотезу необхідно відхилити, а ознаки вважати залежними, оскільки ***p - value <α***.

Завдання

1. Використовуючи тест Шапіро-Уілка, перевірити, чи є нормально розподіленими характеристики квітів ірису (фрейм даних iris). Рівень значимості α= 0.5 .

2. Згенерувати вибірку нормально розподілених випадкових величин з математичним сподіванням k = 10,15,20,25,30 , стандартним відхиленням $\sqrt{2k}$ та вибірку випадкових величин, розподілених за законом $\boldsymbol{\chi^2}$. Розмір вибірок 200 елементів. Використовуючи тест Колмогорова-Смірнова, перевірити гіпотезу про те, що дані вибірок відповідають неперервному розподілу. Рівень значимості α= 0.5.

3. Завантажити з файлу allcountries.txt таблицю в якій міститься інформація про чисельність населення, площі та інші характеристики сучасних держав. Вибрати з таблиці ті країни, для яких доступна інформація про чисельність їх населення (відсутні значення NA) та площа більша 10,00. Нехай ***area_log = log10(log10(area)) population_log = log10 (log10 (population)).*** Побудувати лінійну регресію (використовуючи функцію lm) для залежності ***population _ log*** від ***area _ log*** . Використовуючи тест Колмогорова-Смірнова, перевірити гіпотезу про те, що ***population _ log*** і ***f(area _log),*** є неперервно розподіленими, де ***f ()*** – побудована регресійна функція.

4. Використовуючи критерій $\boldsymbol{\chi^2}$ перевірити нуль-гіпотезу, яка полягає в тому, що колір жіночих очей не залежить від кольору волосся (на фреймі даних ***HairEyeColor***).

5. З файлу Readingspeed.txt завантажити таблицю даних, яка містить інформацію про швидкість читання дітей залежно від методики навчання (DRA – direct reading activities, SC – standart curriculum). Використовуючи t-

тест, перевірити гіпотезу про те, що середній час читання при використанні обох методик навчання співпадає (використовувати різні альтернативні гіпотези). Обґрунтувати отримані результати.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв’язання задач.
5. Висновки.

Контрольні запитання

1. Поясніть призначення функції shapiro.test
2. Назвіть основні параметри функції shapiro.test
3. Поясніть призначення функції ks.test
4. Назвіть основні параметри функції ks.test
5. Поясніть призначення функції t.test
6. Назвіть основні параметри функції t.test.

ЛАБОРАТОРНА РОБОТА № 3

Тема: Задачі відновлення регресії

Мета: Навчитися розв'язувати задачі відновлення регресії за допомогою вбудованих функцій системи R

Теоретичні відомості

Для побудови рівнянь регресії в системі R існує функція lm:

lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)

formula – символьний опис моделі пошуку рівняння регресії;

Формула	*Модель*
`Y ~ A`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A}$
`Y ~ -1 + A`	$\mathtt{Y} = \beta_1\mathtt{A}$
`Y ~ A + I(A^2)`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A} + \beta_2\mathtt{A}^2$
`Y ~ poly(A, 2)`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A} + \beta_2\mathtt{A}^2$
`Y ~ A + B`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A} + \beta_2\mathtt{B}$
`Y ~ A:B`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{AB}$
`Y ~ A*B`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A} + \beta_2\mathtt{B} + \beta_3\mathtt{AB}$
`Y ~ (A + B)^2`	$\mathtt{Y} = \beta_0 + \beta_1\mathtt{A} + \beta_2\mathtt{B} + \beta_4\mathtt{AB}$

Рис. 3. Типи рівнянь регресії

data – назва набору початкових даних. Якщо назва не вказана (або вказано ***NULL***), то всі дані, що зустрічаються у моделі – це вектори робочого простору. В протилежному випадку – це поля вказаного фрейму даних;

subset – вектор, що визначає підмножину даних, які мають місце в моделі. Необов'язкове поле;

weight – вектор вагових коефіцієнтів. Може набувати або числових значень, або значення ***NULL***. Якщо вектор набуває числових значень, то він використовується у методі найменших квадратів. Необов'язковий параметр;

na.action – функція, яка показує, що треба зробити з моделлю, якщо в ній є ***NA***;

method – назва методу, що використовується для відшукання рівняння регресії. На даний час доступний лише метод "qr";

model, x, y, qr – логічні значення. Якщо ***TRUE***, то функція повертає відповідні компоненти моделі: знайдені коефіцієнти β_j , матрицю даних X, стовпчик відповідей у QR-розклад;

singular.ok – якщо ***FALSE***, то виродженість матриці (ранг матриці ***X*** менше числа стовпців) призведе до помилки.

Функція ***lm*** повертає список, який включає поля:

coefficients – вектор коефіцієнтів β_j ;

residuals – вектор різниці $y - X\hat{\beta}$;

rank – ранг матриці ***X***;

fitted.values – обчислені значення $X\hat{\beta}$;

qr – ***QR***-розклад;

qraux – допоміжна інформація, необхідна для відновлення ***QR***-розкладу (для знаходження ***QR***-розкладу використовується функція ***qr***);

pivot – керуючі елементи при відшуканні функції ***qr***;

tol – допустима похибка, що використовується при відшуканні коефіцієнтів рівняння регресії;

model – матриця ***X***.

Приклад 1. Фрейм даних ***airquality*** з пакету ***datasets*** містить щоденні показники якості повітря в Нью-Йорку з травня по вересень 1973 р.: вміст озону Ozone, сонячне випромінювання Solar.R, середня швидкість вітру Wind, максимальна добова температура Temp. Деталі можна проглянути в пакеті даних.

> **Розв'язання:** fit = lm(Ozone ~ ., data = air)

Крок 1. Спочатку намалюємо діаграми розсіювання для кожної пари змінних:

```
> library(datasets)
> air <- airquality[, c("Ozone", "Solar.R", "Wind","Temp")]
> air <- air[!is.na(air$Solar.R) & !is.na(air$Ozone), ]
> pairs(air, panel = panel.smooth, main = "Airquality data", col = "blue")
```

Крок 2. Тепер розглянемо регресійну модель

Ozone = β0 + β1 · Solar.R + β2 · Wind + β3 · Temp

```
> fit = lm(Ozone ~ ., data = air)
  fit
```

```
Call:
lm(formula = Ozone ~ ., data = air)
Coefficients:
(Intercept) Solar.R Wind Temp
-64.34208 0.05982 -3.33359 1.65209
```

Таким чином знайдені наступні значення $\beta 0 = -64.34208$, $\beta 1 = 0.05982$, $\beta 2 = -3.33359$, $\beta 3 = 1.65209$. Більше інформації можна отримати за допомогою функції *summary*:

```
> summary(fit) Call:
lm(formula = Ozone ~ ., data = air)
Residuals:
Min 1Q Median 3Q Max
-40.485 -14.219 -3.551 10.097 95.619
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -64.34208 23.05472 -2.791 0.00623 **
Solar.R 0.05982 0.02319 2.580 0.01124 *
Wind -3.33359 0.65441 -5.094 1.52e-06 ***
Temp 1.65209 0.25353 6.516 2.42e-09 ***
---
Signif. codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1
Residual standard error: 21.18 on 107 degrees of freedom Multiple R-Squared: 0.6059, Adjusted R-squared: 0.5948
F-statistic: 54.83 on 3 and 107 DF, p-value: < 2.2e-16
```

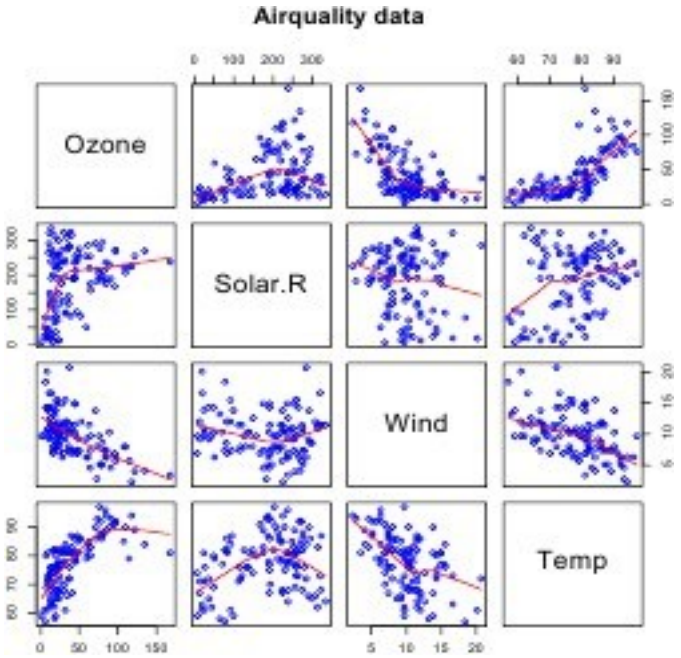

Рис. 4. Діаграми розсіювання для кожної пари змінних

Регуляризацією у статистиці, машинному навчанні та теорії обернених задач називається метод доповнення умови задачі деякою додатковою інформацією з метою розв'язання цієї некоректно поставленої задачі або запобіганню перенавчанню. Ця інформація в більшості випадків набуває вигляду штрафних функції залежно від складності моделі. Наприклад, це можуть бути обмеження гладкості результуючої функції або обмеження по нормі векторного простору.

З точки зору методології регуляризація означає спробу застосувати бритву Оккама до розв'язання задачі. З байєсової точки зору більшість методів регуляризації відповідає доповненню деяких апріорних розподілів на параметри моделі.

Деякі види регуляризації:

- L_1-регуляризація

$$L_1 = \sum_i (y_i - y(t_i))^2 + \lambda \sum_i |a_i|$$

- регуляризація Тихонова для інтегральних рівнянь дозволяє балансувати між відповідністю даних та маленькою нормою розв'язку

$$L_2 = \sum_i (y_i - y(t_i))^2 + \lambda \sum_i {a_i}^2$$

В системі R виконати регуляризацію можна за допомогою функції ***lm.ridge***, що має синтаксис:

lm.ridge(formula, data, lambda = 0,...)

де

formula – символьний опис моделі пошуку рівняння регресії;

data – набір даних для завантаження у формулу;

lambda – параметр регуляризації (скаляр або вектор).

Завдання

1. Завантажити дані з файлу reglab1.txt. Використовуючи функцію **lm**, побудувати всі можливі типи рівнянь регресії. Вибрати найкращу модель, вибір обґрунтувати.

2. Реалізувати алгоритм зменшення кількості ознак, що використовуються для побудови рівнянь регресії, таким чином: для кожного $k \in \{0,1,\ldots,p\}$ визначити підмножину ознак обсягом k, таку, що мінімізує $RSS(\beta)$. За допомогою розроблено алгоритму визначити оптимальну підмножину ознак для даних наведених у файлі reglab2.txt.

Вибір обґрунтувати. Виконати інтерпретацію проведених розрахунків значень t –статистики та $p-value$ для коефіцієнтів $\hat{\beta}$

3. Завантажити дані з файлу cygage.txt. Побудувати рівняння регресії, що відображає залежність віку досліджуваних покладів корисних копалин від глибини їх залягання, використовуючи вагові множники спостережень. Оцінити якість побудованої моделі.

4. Завантажити дані з файлу alligators.txt. визначити найкращу регресійну модель (можливо нелінійну), що відображує залежність ваги алігатора від його довжини.

5. Завантажити бібліотеку ***MASS*** та набір даних ***longley***. Виключити з набору даних змінну ***Population***. Поділити набір даних на тестову та навчальну вибірки однакового обсягу випадковим чином. Побудувати ***ridge regression*** для значень $lambda = 10\hat{}\, seq(-3,2,by=0.2)$ та обчислити помилки на тестовій на навчальній вибірці для заданих значень $lambda$. Побудувати графіки. Пояснити отримані результати.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно варіантом.
4. Протоколи розв'язання задач.
5. Висновки.

Контрольні запитання

1. Поясність суть методу найменших квадратів
2. Парна лінійна регресія
3. Множинна лінійна регресія
4. Тестування і усунення мультиколінеарності
5. Тестування і усунення гетероскедастичності
6. Автокореляція: причини і наслідки

ЛАБОРАТОРНА РОБОТА № 4

Тема: Кластеризація в R

Мета: Навчитися розв'язувати задачі кластеризації за допомогою вбудованих функцій системи R

Теоретичні відомості

Задача кластеризації полягає у визначенні груп об'єктів (процесів), які є найближчими один до іншого за деяким критерієм. При цьому ніяких припущень про їх структуру, як правило, не робиться. Більшість методів кластеризації базується на аналізі матриці коефіцієнтів схожості, до яких належить відстань, кореляція та інші. Якщо критерієм або метрикою виступає відстань, то кластером називають групу точок Ω, таку, що середній квадрат внутрішньо групової відстані до центру групи менше середньої відстані до загального центру в початковому наборі об'єктів, то $d_\Omega^{-2} < d^2$, де $d_\Omega^{-2} = \frac{1}{N}\sum_{x_i \in \Omega}(X_i - \bar{X}_\Omega)^2, \bar{X}_\Omega = \frac{1}{N}\sum_{x_i \in \Omega} X_i$, N –кількість точок в кластері Ω.

Припустимо, що число кластерів K задано і $K<<m$, де m – кількість об'єктів. Отримаємо задачу

$$\sum_{i=1}^{K} \sum_{j=1}^{m_i} \left\| X_j - \bar{X}_i \right\| \to \min \quad (1)$$

де $m_i, i = 1, k$ – кількість об'єктів в і -му кластері, $\bar{X}_i, i = \overline{1, K}$ – середнє значення в кластері, $\left\| X_j - \bar{X}_i \right\|$ – відстань між об'єктами. Розв'язком даної задачі є центри кластерів$\bar{X}_i$, які можуть міститися серед даних об'єктів, що є достатньо строгою умовою, і можуть бути представленими будь-якими точками області дослідження.

До традиційних методів кластерного аналізу, які реалізовані в системі R у вигляді вбудованих функцій, належать **метод k-середніх**, **метод clara** та **метод дендритів**.

Метод k-середніх належить до групи ітеративних методів кластеризації і основі його застосування лежить *базовий алгоритм*:

Крок 1. Емпірично розбити об'єкти на вказане число кластерів. Обчислити центр кожного кластеру.

Крок 2. Помістити кожну точку даних у кластер із найближчим центром ваги.

Крок 3. Обчислити нові центри ваги кластерів. Кластери не змінюються до тих пір, поки не будуть переглянуті всі об'єкти.

Крок 4. Кроки 2 і 3 повторюються до тих пір, поки не перестануть змінюватись кластери.

Існує два основні типи ітерацій: за принципом « k-середніх» і за принципом «сходження на гору». Ітерації за принципом « k-середніх» полягають у переміщенні об'єкта в кластер з найближчим центром ваги. Вони можуть бути комбінаторними або некомбінаторними. У першому випадку перерахунок центра кластеру здійснюється після кожної зміни його складу, в іншому – лише після того, як буде завершено перегляд усіх даних. Крім того, ітерації за принципом « k -середніх» є включаючими та виключаючими. У включаючих ітераціях після обчислення центру кластера об'єкт включається до складу кластера, у виключаючих – вилучається із кластера. В ітераціях, що реалізуються за принципом «сходження на гору», переміщення об'єктів відбувається, виходячи із того, чи буде таке переміщення оптимізувати значення деякого статистичного критерію.

До функцій, що визначають якість кластеризації (статистичні критерії), належать trW, $trW^{-1}B$, $\det W$ та найбільше власне число матриці $W^{-1}B$, де W – об'єднана внутрішньо-групова коваріаційна матриця, В – об'єднана міжгрупова коваріаційна матриця. Використовуючи кожний із статистичних критеріїв знаходять кластери визначеного вигляду. Так, критерій *trW* орієнтований на утворення гіперсферичних однорідних кластерів. За критерієм *detW* передбачається, що у кластерів буде однакова форма, не обов'язково гіперсферична.

В системі R для групування об'єктів за методом **k-середніх** існує функція

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))

де ***x*** – набір початкових даних;

centers – кількість кластерів або множина початкових центрів кластерів. Якщо аргумент є числом, то множина центрів кластерів генерується випадковим чином.

iter.max – максимальна кількість ітерацій;

nstart – якщо centers, то аргумент визначає число випадково обраних множин;

algorithm – назва алгоритму;

повертає список, який містить поля:

cluster – вектор цілих чисел, який вказує на належність об'єктів кластерам;

centers – матриця центрів кластерів;

withinss – сума квадратів відстаней між точками у кожному кластері;

size – кількість точок у кожному кластері.

Приклад 1. Згенеруємо випадкові дані, які поділимо на 2 кластери з використанням методу **kmeans** (Рис. 5-6) **.**

Розв'язання:

```
> x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2), matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
> colnames(x) <- c("x", "y")
> cl <- kmeans(x, 2)
>plot(x, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 8, cex=2)
```

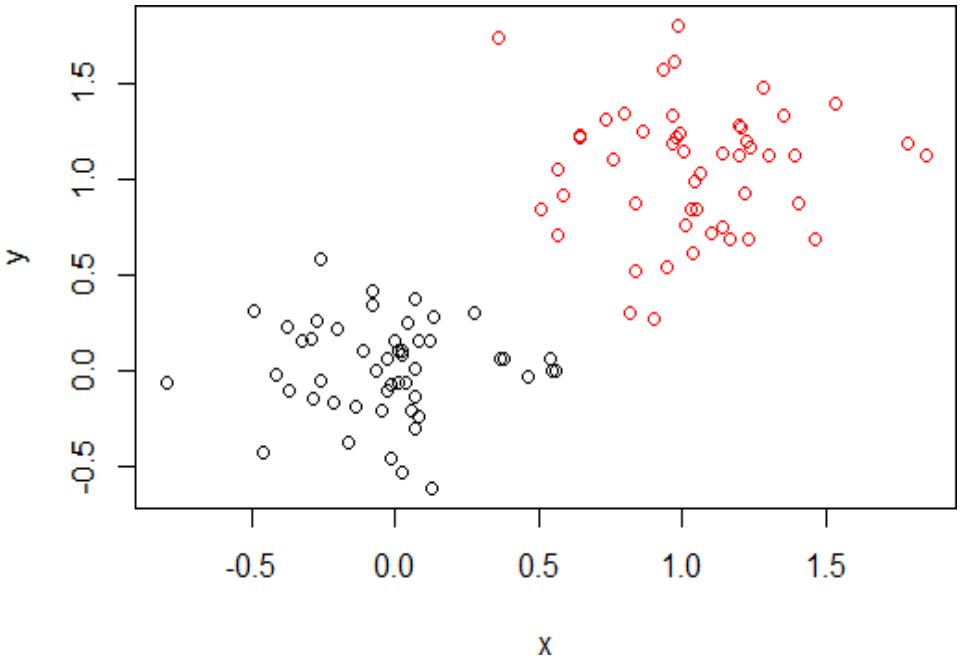

Рис. 5. Графічна інтерпретація результатів поділу множини точок на 2 кластери

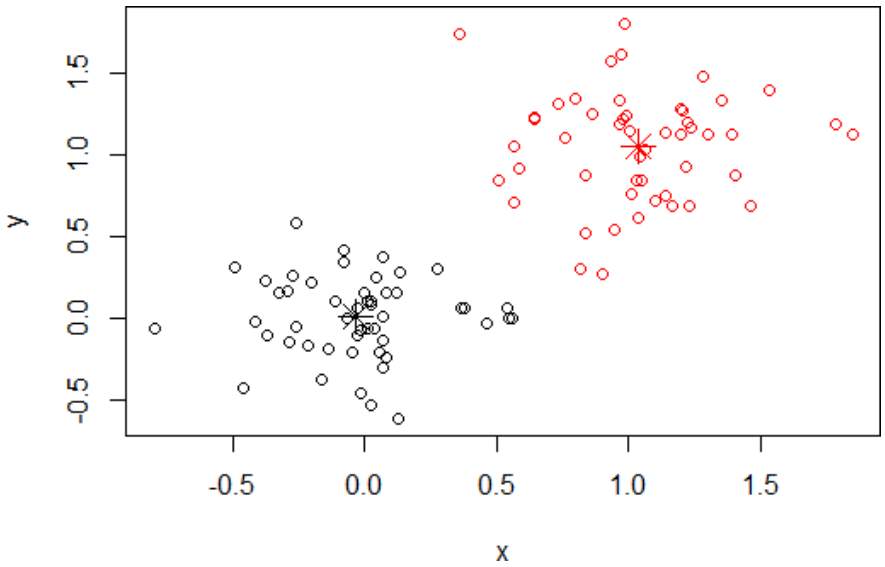

Рис. 6.– Графічна інтерпретація результатів поділу
множини точок на 2 кластери із зазначенням центрів кластерів

Алгоритм CLARA (Clustering Large Applications) був розроблений у 1990 році дослідниками Лео Кауфманом (Leo Kaufman) і Пітером Дж. Руссо (Peter J. Rousseeuw).

Вони описали цей алгоритм у своїй книзі “Finding Groups in Data: An Introduction to Cluster Analysis”. CLARA є вдосконаленням алгоритму PAM (Partitioning Around Medoids) і призначений для роботи з великими наборами даних, де PAM виявляється обчислювально складним через високу вимогливість до пам’яті й процесорного часу.

Ефективність алгоритму залежить від обраного для зразка набору даних. Як правило, прийнятні результати кластеризації на обраному набор даних можуть не дати прийнятних результатів кластеризації на всьому наборі даних.

В системі R для групування об’єктів за методом **clara** існує функція

clara(x, k, metric = "euclidean", stand = FALSE, samples= 5,
sampsize = min(n 40 + 2* k), trace=0,medoids.x = TRUE,
keep.data = medoids.x, rngR = FALSE)

де

x – матриця початкових даних (або фрейм), кожному рядку якої відповідають спостереження, а стовпцям – назви змінних. Всі змінні повинні мати числовий формат. Відсутність значень (***NA***) не допускається;

k – кількість кластерів, ***0*** □ ***k*** □ ***n***,

n – кількість об’єктів;

metric – символьний опис метрики, яка використовується для обчислення відстані між об'єктами. Доступні значення "***euclidean***" – відстань Евкліда, "***manhattan***" – манхетенська відстань;

stand – логічна змінна, яка вказує на стандартизацію даних. Суть стандартизації полягає в обчисленні для кожної змінної відношення математичного сподівання до стандартного відхилення;

samples – число семплів (додаткових наборів), які добуваються з початкового набору даних;

sampsize – число спостережень в кожному семплі (додатковому наборі). Значення ***Sampsize*** має бути більшим ніж число кластерів, але меншим ніж кількість об'єктів;

trace – ціле число, яке позначає номер ітерації алгоритму, починаючи з якої необхідно виводити результати;

modoids.x – логічна змінна в якій зазначається необхідність виведення медіани. Якщо змінна приймає значення FALSE, то алгоритм повертає лише номери рядків матриці x, які містять медіани;

keep.data – логічна змінна в якій зазначається необхідність збереження початкових даних;

rngR – логічна змінна, що має місце тоді, коли замість вбудованого генератора випадкових чисел функції ***clara()***, використовується вбудований генератор випадкових чисел системи R.

повертає список, який містить поля:

cluster – вектор цілих чисел, який вказує на належність об'єктів кластерам;

centers – матриця центрів кластерів;

withinss – сума квадратів відстаней між точками у кожному кластері;

size – кількість точок у кожному кластері.

sample – номери об'єктів, які ввійшли до семплу;

medoids – медіана кожного кластеру;

med – індекси медіан кластерів (***medoids=x[i.med,]***)

clustering – об'єкт класу ***partition.object***;

objective – цільова функція остаточного поділу на кластери початкового набору даних;

clusinfo – матриця кожний рядок якої містить чисельну інформацію про кластер: кількість об'єктів в кластері, максимальну та середню різницю об'єктів кластеру, медіану кластеру. Останній стовпчик – це максимальна

різниця між об’єктами кластеру та медіаною кластеру, що поділена на мінімальну різницю між медіаною кластеру та медіанами інших кластерів. Чим менше значення цього відношення, тим якіснішою є кластеризація;

diss – відмінність;

silinfo – інформація про ширину найкращого семплу;

call – виклик функції;

data – матриця даних.

Приклад 2. Фрейм даних ruspini з пакету cluster, що містить координати 75 точок на площині, необхідно поділити на 4 кластери.

Розв’язання:

```
> library(cluster)
> cl=clara(ruspini,4)
> plot(ruspini, col=cl$clustering, xlab='x', ylab='y')
```

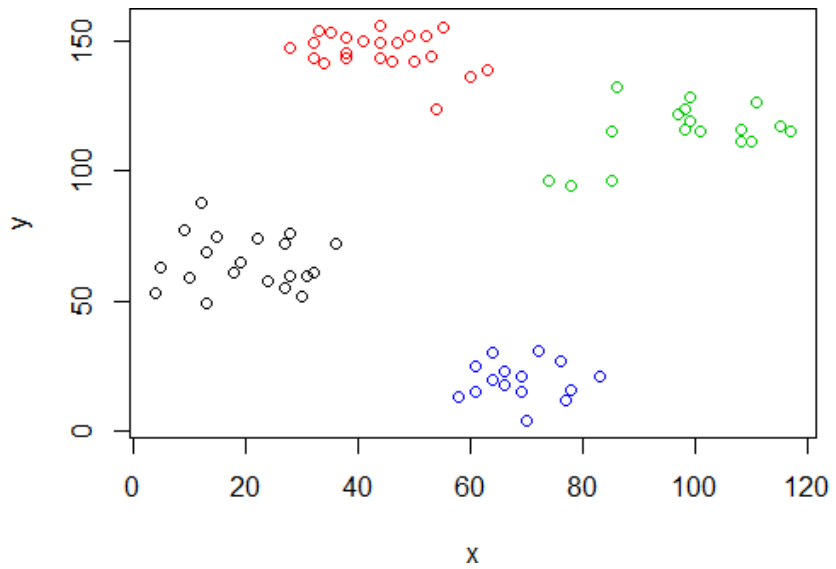

Рис. 7. Графічна інтерпретація результатів поділу множини точок на 4 кластери

Спробуємо поділити цю множину точок на 5 кластерів.

```
 > cl=clara(ruspini,5)
>plot(ruspini, col=cl$clustering, xlab='x', ylab='y')
```

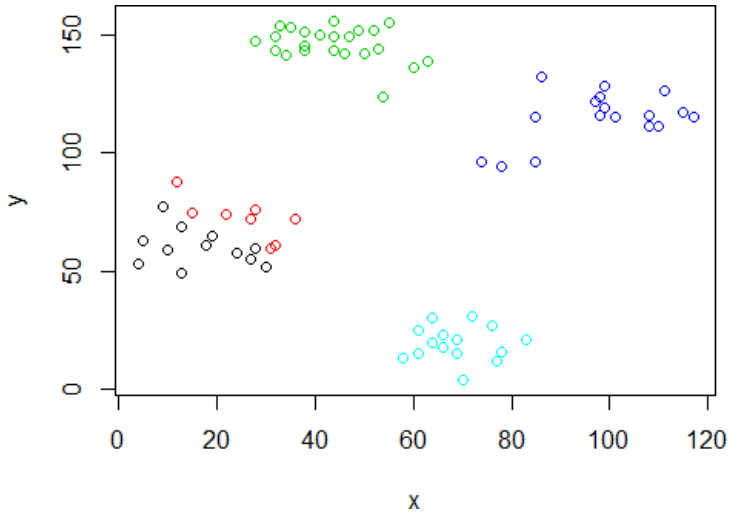

Рис. 8. Графічна інтерпретація результатів поділу множини точок на 5 кластери

Дендрограмою зазвичай називається дерево, тобто граф без циклів, який будується на основі матриці подібностей між об’єктами.

Основними методами побудови дендрограм є:

1. Метод одиночній зв'язку (англ. single linkage) . Також відомий, як «метод найближчого сусіда».
2. Метод повного зв'язку (англ. complete linkage). Також відомий, як «метод далекого сусіда».
3. Метод середнього зв'язку (англ. pair – group method using arithmetic averages).

Незважений (англ. unweighted) та зважений (англ. weighted) .

4. Центроїдний метод (англ. pair – group method using the centroid average) .

Незважений .та зважений (медіанний) .

5. Метод Уорда (англ. Ward's method).

В системі R для побудови **дендрограм** існує функція

agnes(x, diss = inherits(x, "dist"), metric = "euclidean", stand = FALSE, method = "average", par.method, keep.diss = n < 100, keep.data = !diss)

де

x – матриця початкових даних або набір даних (або матриця відмінностей) залежно від значення аргумента ***diss***. Якщо ***x*** – матриця початкових даних або набір даних, то кожному рядку відповідають об’єкти, а стовпцям – назви змінних. Всі змінні повинні мати числовий формат. Відсутність значень (***NA***) не допускається;

diss – логічна функція, яка вказує на спосіб поділу матриці ***x***. Значення ***TRUE*** відповідає тому, що *x* буде розглядатися як матриця відмінностей, а ***FALSE*** – як матриця об'єктів;

metric – символьний опис метрики, яка використовується для обчислення відстані між об'єктами. Доступні значення "***euclidean***" – відстань Евкліда, "***manhattan***" – манхетенська відстань; ***stand*** – число семплів (додаткових наборів), які добуваються з початкового набору даних;

stand – логічна змінна, яка вказує на стандартизацію даних. Суть стандартизації полягає в обчисленні для кожної змінної відношення математичного сподівання до стандартного відхилення;

method – символьний рядок в якому зазначається назва методу кластеризації. Доступно шість методів: «***average***» (агломеративний метод усереднення),

«***single***» (алгоритм «простого зв'язку»), «***complete***» (алгоритм «повного зв'язку»), «***ward***» (метод Варда), «***weighted***» (зважений усереднений агломеративний метод), «***flexible***» (узагальнений метод).

par.method – якщо ***method=="flexible",*** вектор чисел довжини 1,3 або 4, що містить параметри функції Ланса-Вільямса;

повертає список, який містить поля:

order – вектор, що містить впорядкування оригінальних об'єктів, і побудований такими чином, щоб гілки дендрограми не перетиналися;

order.lab – вектор, що містить позначення об'єктів, а не їх номери;

height – вектор відстаней між об'єднаними кластерами;

ac – агломеративний коефіцієнт, значення якого показує кластерну структуру набору даних;

diss – об'єкт класу ***dissimilarity,*** який являє собою матрицю відмінностей між початковими даними;

data – матриця початкових або стандартизованих даних (залежно від значення аргументу ***stand***).

Приклад 3. Побудуємо дендрограму для набору даних ***agriculture*** в якому міститься інформація про частку населення зайнятого в сільському господарстві та валових національний продукт на душу населення певних країн світу

```
>library(cluster)
>data(agriculture)
>agriculture
    > x     y
 B   16.8  2.7
 DK  21.3  5.7
 D   18.7  3.5
 GR   5.9 22.2
 E   11.4 10.9
 F   17.8  6.0
 IRL 10.9 14.0
 I   16.6  8.5
 L   21.0  3.5
 NL  16.4  4.3
 P   7.8 17.4
 UK  14.0  2.3
 >       plot(agnes(agriculture))
```

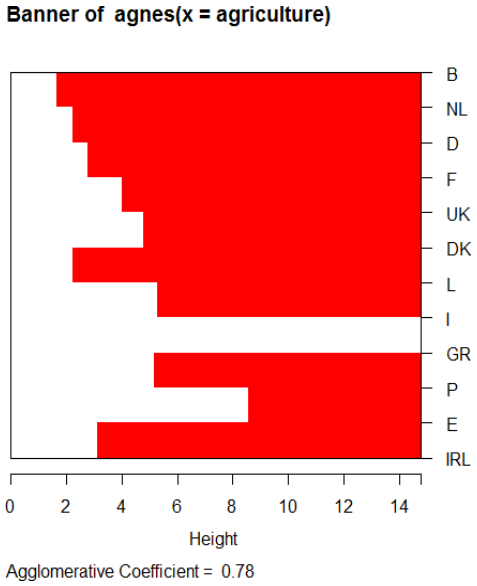

Рис. 9. Гістограма набору даних agriculture

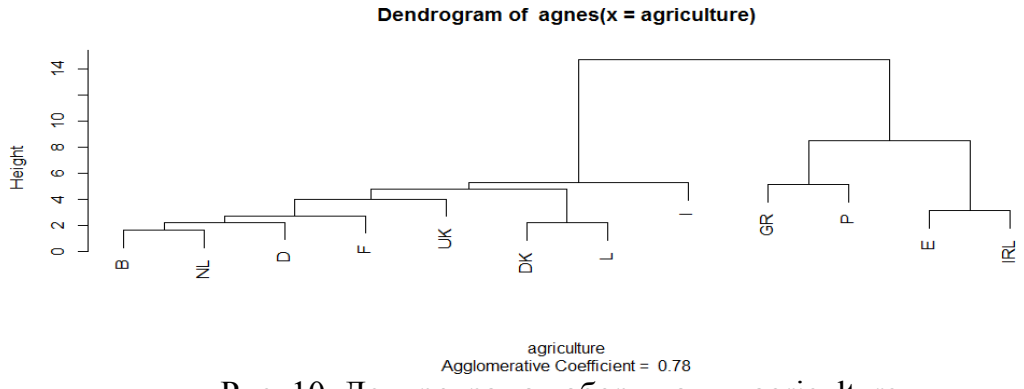

Рис. 10. Дендрограма набору даних agriculture

З рис. 10 видно, що країни в яких спостерігається найбільша частка сільського господарства об'єднані в один кластер (Греція, Португалія, Іспанія та Ірландія).

Завдання

1. Множину об'єктів з набору даних ***mtcars*** поділити на 3 кластери методом центру ваги (***kmeans***). Дослідити як змінюється якість кластеризації із зростанням кількості ітерацій алгоритму.

2. Згенерувати набір даних у двовимірному просторі, який складається з 3 кластерів, кожний з яких сильно «витягнутий» вздовж однієї з осей. Дослідити якість кластеризації методом ***clara*** при 1) використанні стандартизації; 2) залежно від типу метрики. Обґрунтувати отримані результати.

3. Побудувати дендрограму для набору даних ***USArrests*** (набір містить статистику про кількість арештів за різні види злочинів у кожному зі штатів США, статистика за 1973 р.). Інтерпретувати отримані результати.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв'язання задач.
5. Висновки.

Контрольні запитання

1. Що таке однорівнева і ієрархічна кластеризація?
2. Що таке чітка і нечітка кластеризація?
3. Які є підходи до розрахунку відстані між кластерами?
4. Поясніть різницю між алгомеративною і дивізимною ієрархічними кластеризаціями?
5. Опишіть метод кластеризації к-середніх.

ЛАБОРАТОРНА РОБОТА № 5

Тема: Створення графічного інтерфейсу користувача в середовищі RStudio з використанням Shiny

Мета: Навчитися створювати власні інтерактивні додатки, що дозволяють завантажувати дані, відображати їх у таблицях та графіках, а також виводити результати аналізу.

Теоретичні відомості

Shiny — це фреймворк на мові R, який дозволяє створювати інтерактивні веб-додатки без необхідності знання HTML, CSS чи JavaScript. Він розроблений компанією Posit (RStudio) і використовується для:

- побудови графічних інтерфейсів до R-програм;
- створення інтерактивних звітів і панелей управління (dashboard);
- організації візуалізації результатів обчислень у режимі реального часу;
- швидкого прототипування аналітичних інструментів.

Shiny-додатки можуть працювати:

- локально в RStudio (натисненням Run App);
- через сервер shinyapps.io або власний Shiny Server.

Будь-який Shiny-додаток складається з двох основних частин:

1. **Інтерфейс користувача (UI)** — описує, як виглядає програма:

```
ui <- fluidPage(
  titlePanel("ShinyApp для додаткових балів"),
  sidebarLayout(sidebarPanel(sliderInput("n", "Кількість точок:", 10, 100, 50)),
    mainPanel(plotOutput("p"))))
```

2. **Серверна частина (Server)** — описує, як програма реагує на дії користувача:

```
server <- function(input, output) {
  output$p <- renderPlot({
    x <- rnorm(input$n)
    hist(x, col = "skyblue", main = "Гістограма випадкових даних")})
}
```

Основні елементи інтерфейсу:

***titlePanel()* —** Використовується для створення заголовка програми. У верхній частині вікна з'являється великий напис із назвою додатку.

sidebarLayout(), sidebarPanel(), mainPanel()* -** Ці функції формують основну структуру сторінки: ***sidebarLayout() ділить простір на дві частини - ліву панель (керування) та праву панель (результати); ***sidebarPanel()*** містить елементи введення даних до яких належать кнопки, поля, випадаючі списки; ***mainPanel()*** містить елементи виведення - таблиці, графіки, текстові результати.

tabsetPanel() та ***tabPanel()* —**Служать для створення вкладок у головному вікні: ***tabsetPanel()*** — контейнер, який містить кілька вкладок; ***tabPanel()*** — окрема вкладка, в якій розміщується свій вміст наприклад, таблиця чи графік.

***fluidPage()* —** Є базовим контейнером усієї сторінки Shiny-додатку та забезпечує адаптивну структуру, яка автоматично підлаштовується під розмір вікна.

***h1() … h5()* —** Це функції для створення заголовків різних рівнів, вони застосовуються для структурування тексту або відділення логічних блоків у додатку.

***actionButton()* —**Створює інтерактивну кнопку, натискання якої запускає певну дію у серверній частині програми.

***fileInput()* —** Дозволяє завантажити файл із локального комп'ютера.

***selectInput()* —** Створює випадаючий список для вибору змінної, параметра або опції.

***checkboxInput()* —** Створює перемикач, що приймає два стани TRUE або FALSE.

***numericInput()* —** Створює поле для введення числового значення.

***plotOutput()* —** Використовується для виведення графіків, створених у серверній частині.

***verbatimTextOutput()* —** Виводить текстову або числову інформацію у форматі «як є».

Основні функції серверної частини:

renderPlot()* —** створює графік для виведення в інтерфейс**,** використовується разом із ***plotOutput() у частині ***ui***. Дозволяє будувати як базові графіки, так і з бібліотеки **ggplot2**.

renderText() **—** виводить текстовий результат та застосовується для відображення динамічних повідомлень або коротких пояснень.

renderPrint() — виводить текст у форматі консольного результату. Підходить для відображення результатів функцій ***summary(), str(), lm(), kmeans()*** тощо.

renderTable() — створює просту таблицю з даних.

DT::renderDT() **—** створює інтерактивну таблицю яка забезпечує сортування, фільтрування, пагінацію. Використовується разом із ***DT::datatable()*** у інтерфейсі.

observeEvent(input$...) — виконує певну дію після натискання кнопки або зміни параметра.

reactiveVal() **—** створює змінну, значення якої можна змінювати динамічно і застосовується для зберігання даних або результатів, які оновлюються під час роботи додатку.

reactive() **—** створює реактивний об’єкт, який автоматично оновлюється при зміні залежних значень. Використовується для попередньої обробки або фільтрації даних.

req() — зупиняє виконання коду, якщо умова не виконується або дані ще не завантажені.

validate() і ***need()*** — використовуються для перевірки правильності введених даних і відображення повідомлень про помилки.

showNotification() — показує коротке повідомлення користувачу у правому верхньому куті екрану.

updateSelectInput() — оновлює елементи керування під час роботи програми.

output$... **—** об’єкти для відображення результатів у UI. Кожен елемент ***output*** у серверній частині відповідає елементу в інтерфейсі з таким самим ідентифікатором (***outputId***).

shinyApp(ui, server) **—** Це заключна команда, яка поєднує інтерфейс та логіку сервера в єдиний Shiny-додаток.

Приклад 1. Створіть базовий інтерфейс Shiny-додатка з панеллю керування та двома вкладками: «Дані» і «Графік». Додайте заголовок сторінки.

Розв'язання:

```
library(shiny)
ui <- fluidPage(
titlePanel("Мій перший Shiny-додаток"),
sidebarLayout(
sidebarPanel(h4("Панель керування")),
   mainPanel(tabsetPanel(tabPanel("Дані"), tabPanel("Графік")))
  )
)
server <- function(input, output) {}
shinyApp(ui, server)
```

Приклад 2. Завантажте таблицю з **.txt** файлу, у форматі перший рядок заголовки та роздільник між стовпцями табуляція і відобразіть її.

Розв'язання:

```
library(shiny)
ui <- fluidPage(
fileInput("file", "Завантажити .txt", accept = ".txt"),
tableOutput("data"))
server <- function(input, output) {
output$data <- renderTable({
req(input$file)
read.table(input$file$datapath, header = TRUE, sep = "\t")})
}
shinyApp(ui, server)
```

Приклад 3. Створіть Shiny-додаток, який будує дендрограму для набору mtcars. Програма має стандартизувати дані, обчислити матрицю відстаней і виконати ієрархічну кластеризацію методом complete.

Розв'язання:

```
library(shiny)
ui <- fluidPage(
titlePanel("Дендрограма для mtcars"),
mainPanel(plotOutput("dend")))
server <- function(input, output) {
output$dend <- renderPlot({
data <- scale(mtcars)
dist_mat <- dist(data)
hc <- hclust(dist_mat, method = "complete")
plot(hc, main = "Ієрархічна кластеризація",
xlab = "Автомобілі", sub = "", cex = 0.6)
rect.hclust(hc, k = 4, border = 2:5)})}
shinyApp(ui, server)
```

Приклад 4. Створіть додаток, який дозволяє завантажити .txt-файл, обрати дві числові змінні й отримати коефіцієнт кореляції Пірсона між ними. Додаток повинен відображати числове значення коефіцієнта та текстову інтерпретацію зв'язку.

Розв'язання:

```
library(shiny)
ui <- fluidPage(
titlePanel("Кореляційний аналіз"), sidebarLayout( sidebarPanel(
fileInput("file", "Завантажити .txt", accept = ".txt"),
 uiOutput("var_select"),
actionButton("calc", "Обчислити кореляцію")    ),
mainPanel(verbatimTextOutput("result"))))
server <- function(input, output) {
data <- reactive({req(input$file)
read.table(input$file$datapath, header = TRUE, sep = "\t")  })
output$var_select <- renderUI({req(data())
nums <- names(data())[sapply(data(), is.numeric)]
tagList( selectInput("x", "Перша змінна:", nums), selectInput("y", "Друга змінна:",
nums))  })
observeEvent(input$calc, {req(input$x, input$y)
df <- data()
r <- cor(df[[input$x]], df[[input$y]], use = "complete.obs")
level <- ifelse(abs(r) > 0.7, "сильний", ifelse(abs(r) > 0.4, "помірний", "слабкий"))
dir <- ifelse(r > 0, "прямий", "зворотний")
output$result <- renderPrint({
cat("Коефіцієнт кореляції:", round(r, 3), "\n", "Зв'язок:", level, dir)    })  })}
shinyApp(ui, server)
```

Завдання

1. Створіть основний інтерфейс програми. За допомогою функцій ***fluidPage(), sidebarLayout(), sidebarPanel()*** та ***mainPanel()*** побудуйте базову структуру сторінки з елементами керування та основною областю виведення. Передбачте заголовок програми та адаптивне відображення елементів у вікні користувача.

2. Реалізуйте елементи взаємодії з користувачем. Додайте в інтерфейс елементи введення даних, такі як ***fileInput(), textInput(), numericInput()*** або ***selectInput()***. Реалізуйте кнопки для виконання дій за допомогою ***actionButton()***. Забезпечте відображення текстової або табличної інформації у головній області за допомогою ***renderText(), renderTable()*** або ***DT::renderDT()***.

3. Створіть область для графічного відображення результатів. Реалізуйте відображення графіків за допомогою ***plotOutput()*** і ***renderPlot()***. Передбачте вибір змінних або параметрів для побудови графіка через елементи введення ***selectInput(), sliderInput()*** тощо.

4. Додайте блок для виведення результатів обчислень. Реалізуйте текстовий вихід для показу статистичних показників, повідомлень або результатів аналізу. Для цього використайте ***verbatimTextOutput()*** у поєднанні з ***renderPrint()*** або ***renderText()***.

5. Інтегруйте попередню роботу або аналітичний модуль у створений інтерфейс. Додайте окрему вкладку або розділ, у якому буде реалізовано один із раніше виконаних аналітичних методів, наприклад, кластеризацію, регресійний аналіз, перевірку гіпотез або статистичний тест. Передбачте можливість введення параметрів користувачем, запуску аналізу кнопкою ***actionButton()*** і відображення результатів у вигляді графіків та текстових повідомлень.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв’язання задач.
5. Висновки.

Контрольні запитання

1. Що таке Shiny та для чого його використовують у середовищі RStudio?
2. Яку роль відіграють елементи ui та server у структурі Shiny-додатка?
3. Які функції використовуються для створення елементів введення даних у Shiny?
4. Як відбувається обробка завантажених користувачем даних у Shiny-додатку?
5. Яким чином у Shiny реалізується побудова графічних візуалізацій?

ЛАБОРАТОРНА РОБОТА № 6

Тема: Побудова дерев рішень

Мета: Навчитися розв’язувати задачі класифікації за допомогою вбудованих функцій системи R

Теоретичні відомості

Дерева рішень (дерева вирішальних правил) – один з методів автоматичного аналізу даних, що задає спосіб подання правил виду «Якщо – то» в ієрархічній послідовній структурі, де кожному об’єкту відповідає єдина вершина, що дає рішення.

Основними поняттями теорії дерев рішень, є:

– об’єкт – приклад, шаблон, спостереження;

– атрибут – ознака, незалежна змінна, властивість;

– мітка класу – залежна (цільова) змінна, ознака, що визначає клас об’єкта;

– вузол (вершина) – внутрішній вузол дерева, вузол перевірки;

– лист – кінцевий вузол дерева, вузол рішення;

– перевірка (test) – умова у вузлі.

Дерева рішень запропоновані П. Ховлендом (P. Hoveland) та Е. Хантом (E. Hunt) наприкінці 50-х років XX століття.

Структура дерева рішень наведена на рис. 11.

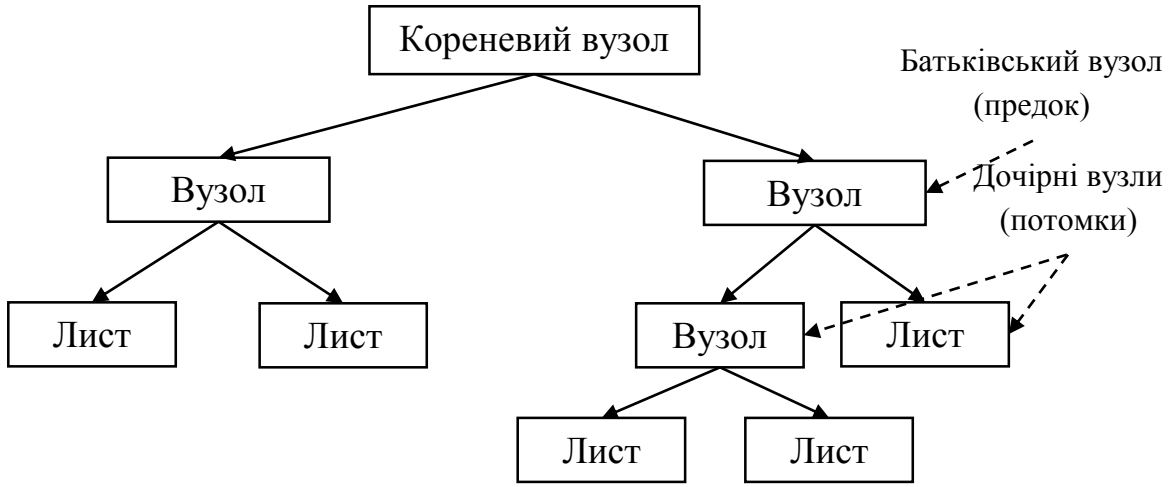

Рис. 11. Вузли (вершини) та листя дерева рішень

Для роботи з деревами рішень в системі R існують функції ***tree***, ***snip.tree*** та ***prune.tree***.

Функція ***tree*** призначена для побудови класифікаційного дерева і має наступний синтаксис

tree(formula, data, weights, subset,
na.action = na.pass,
control = tree.control(nobs, ...),
method = "recursive.partition", split = c("deviance", "gini"),
model = FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)

де

formula – тип формули;

data – набір даних до якого застосовується ***formula, weights*** та ***subset***;

weights – вектор вагових множників прецедентів, який може набувати лише додатних значень;

subset – вираз, який вказує на підмножину прецедентів, що необхідно використати;

na.action – функція, призначена для відновлення пропущених даних в навчальній вибірці. За промовчуванням використовується функція ***na.pass*** (нічого не робити). При цьому пропущені значення будуть відновлені в побудованому дереві і розташовуватимуться якнайглибше;

control – список, який повертає функція ***tree.control***;

method – символьний рядок, що вказує на використовуваний метод. Єдине інше корисне значення ***'model.frame'***;

split – критерій поділу;

model – якщо цей аргумент являє собою дані моделі, тоді ***formula*** і ***data*** аргументи ігноруються, а ***model*** використовується для того, щоб визначити модель. Якщо аргумент є логічною змінною і дорівнює ***TRUE***, тоді дані моделі зберігаються як компонент ***model*** у результаті;

x – булева змінна. Якщо дорівнює ***TRUE***, то для кожного прецеденту повертається матриця змінних;

y – булева змінна. Якщо дорівнює ***TRUE***, то повертається змінна-відповідь;

wts – булева змінна. Якщо ***TRUE***, то повертаються значення вагових множників;

... додаткові аргументи або функції, що передаються;

tree.control. В більшості випадків використовуються для ***mincut***, ***minsize*** або ***mindev***.

повертає список, який містить поля:

frame – набір даних, який для кожної вершини містить такі рядки:

row.names – номер вершини. В стовпцях зазначається значення змінної ***var,*** яка використовується для поділу (або ***<leaf>*** для термінальної вершини), ***n*** – зважене число прецедентів, які відповідають заданій вершині, ***dev – deviance*** даної вершини, ***yval*** – скореговане значення у вершині (математичне сподівання для задач регресійних дерев, переважаючий клас для задач класифікації), ***split*** – матриця індексів для лівого та правого нащадків даного вершини, ***yprob*** – матриця скорегованих ймовірностей для всіх можливих відповідей;

where – вектор цілих чисел, який вказує на приналежність прецедентів навчальної вибірки певним вершинам

terms – змінні формули;

call – the matched call to Tree;

model – якщо ***model = TRUE***, дані моделі;

x – якщо ***x = TRUE***, матриця, що відповідає побудованій моделі;

y – якщо ***y = TRUE***, відповідь;

wts – якщо ***wts = TRUE***, вагові множники;

xlevels – атрибути;

ylevels – атрибути класифікаційних дерев;

singlenode – клас дерев без поділу.

Приклад 1. Завантажити набір даних ***monica*** з пакету ***DAAG*** та побудувати класифікаційне дерево для моделі типу ***outcome~.***

Розв'язання:

Крок 1. Встановити бібліотеку ***DAAG***. Для цього перейти до меню ***Packages*** головного меню системи R. У списку, що з'явиться обрати ***Install package(s).*** У вікні ***CRAN*** вказати локальну належність та натиснути кнопку **ОК**. Після виконання описаних дій на екрані з'явиться вікно ***Packages*** з переліком бібліотек, які можна встановити. Клікнути назву бібліотеки ***DAAG*** та натиснути кнопку **ОК**.

Крок 2. Встановити бібліотеку ***tree.***

Крок 3. Встановити бібліотеку **maptree.**

Крок 4. Завантажити бібліотеку ***DAAG***.

```
> library(DAAG)
```

Крок 5. Завантажити бібліотеку ***tree***.

```
>     library(tree)
```

Крок 6. Завантажити бібліотеку ***maptree***.

```
>     library(maptree)
```

Крок 7. Завантажити набір даних ***monica***.

```
>   data(monica)
```

Крок 8. Побудувати класифікаційне дерево для набору даних ***monica.***

```
>   mon.tr<-(tree(outcome ~.,monica))
```

Крок 9. Побудувати графічне зображення дерева за допомогою функції ***plot***.

```
>   plot(mon.tr,type="uniform")
>   text(mon.tr)
```

Крок 10. Розфарбувати вершини дерева використовуючи функцію draw.tree з бібліотеки **maptree.**

```
>   draw.tree(mon.tr)
```

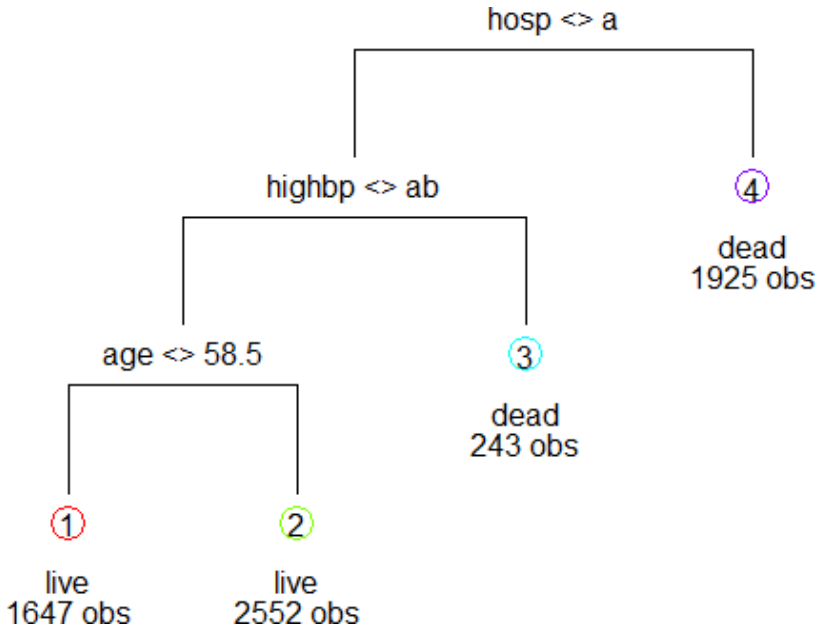

Рис. 12. Класифікаційне дерево набору даних ***monica***

Функція ***snip.tree*** призначена для спрощення класифікаційного дерева шляхом відтинання. Вона має наступний синтаксис:

snip.tree(tree, nodes, xy.save = FALSE,
digits = getOption("digits") - 3)

де

tree – об'єкт класу ***tree***;

nodes – вектор цілих чисел, що містить номери кореневих вершин піддерев, які необхідно відітнути. Якщо аргумент не задано, то користувач має вибрати вершину починаючи з якої має бути видалена частина дерева;

xy.save якщо аргумент дорівнює ***TRUE***, то x та y інтерактивно обрані координати, які зберігаються в об'єкті ***.xy*** в глобальному середовищі;

digits – точність (кількість цифр), яка використовується для виведення статистичних даних про обрані вершини;

Об'єкт класу ***tree***, повертає номери вершин, які залишилися після того, як задані (вибрані підддерева) були видалені.

Визначає ***nested*** послідовність піддерев заданого дерева шляхом рекурсивного видалення (відтинання) найменш важливих поділів.

Приклад 2. Необхідно з'ясувати чи є побудоване в прикладі 1 дерево надлишковим. Якщо так, то виконати операцію «***snip off***» над відповідними вузлами.

Розв'язання. Виконаємо відтинання дерева, що наведено на рис. 12 у вершині з номером 4, а результат помістимо в mon.tr1 (рис. 13).

```
> mon.tr1<-snip.tree(mon.tr,nodes=4)
> draw.tree (mon.tr1)
> mon.tr1
```

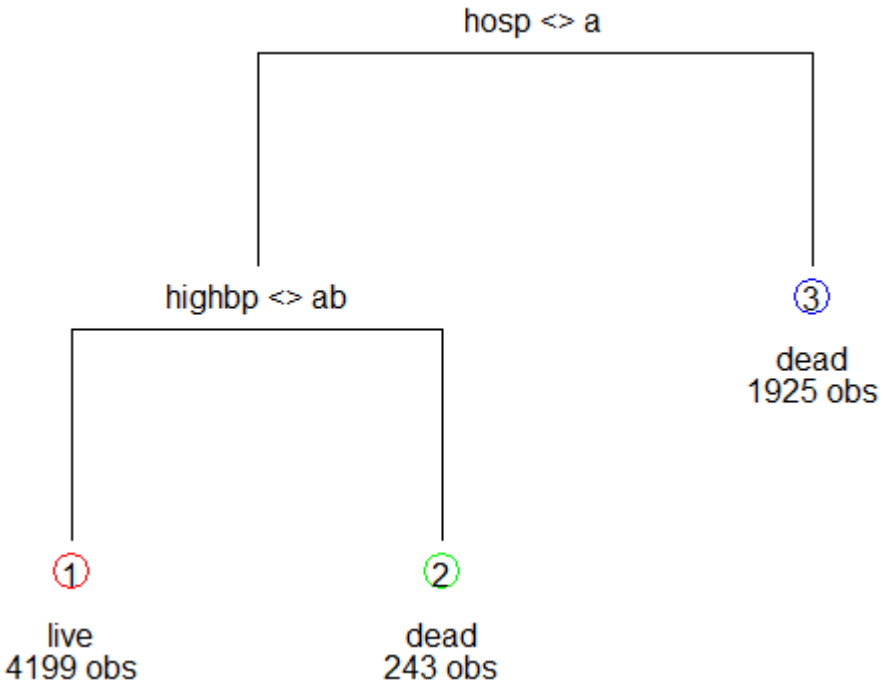

Рис. 13. Класифікаційне дерево після відтинання

Функція ***prune.tree*** призначена для побудови регресійного дерева і має такий синтаксис:

prune.tree(tree, k = NULL, best = NULL, newdata, nwts,
method = c("deviance", "misclass"), loss, eps = 1e-3)
prune.misclass(tree, k = NULL, best = NULL, newdata,
nwts, loss, eps = 1e-3)

де

tree – навчальна модель класу ***tree***;

k – параметр ***cost-complexity***, який вказує на піддерево заданого дерева (якщо ***k*** скаляр) або (додатково) послідовність піддерев, які мінімізують (***cost- complexity measure***) (якщо ***k*** вектор). Якщо параметр не задано, то він автоматично визначається алгоритмом;

best – ціле число, яке «пропонує» розмір заданого піддерева в ***cost-complexity*** послідовності, які має бути виведена;

newdata – дані, які використовуються для обчислення послідовності ***cost-complexity*** піддерев. Якщо користувач не зазначить значення аргументу, то алгоритм використає ті самі значення, що і при побудові дерева;

nwts – вагові множники прецедентів з ***newdata***;

method – символьний рядок, який вказує на функцію вимірювання неоднорідності вершин, які використовуються для ***cost-complexity*** відтинання; ***loss*** – матриця втрат: визначає для кожного класу чисельне значення втрат в разі допущення помилки;

eps – нижня границя ймовірностей, які використовуються для обчислення ***deviances***;

повертає список, який містить поля:

size – число термінальних вершин в кожному дереві cost-complexity pruning послідовності;

deviance – сумарна ***deviance*** кожного дерева ***cost-complexity pruning*** послідовності;

k – значення ***cost-complexity pruning*** параметру кожного дерева в послідовності.

Завдання

1. Завантажити набір даних ***monica*** з пакету ***DAAG***. Побудувати класифікаційне дерево для моделі типу ***outcome~.*** Пояснити отримані результати. Визначити чи є побудоване дерево надлишковим. Якщо так, то виконати операцію «***snip off***» над відповідними вузлами.

2. Завантажити набір даних ***spam7*** з пакету ***DAAG***. Побудувати класифікаційне дерево для моделі типу ***yesno~.*** Пояснити отримані результати. Запустити процедуру «***costcomplexity pruning***» з вибором параметру ***k*** за промовчування, ***method='misclass'*** вивести отриману послідовність дерев. Визначити оптимальне дерево. Вибір оптимального дерева обґрунтувати.

3. Завантажити набір даних ***nsw74psid1*** з пакету. Побудувати регресійне дерево для моделі типу ***re78~***. Побудувати регресійну модель та SVM-регресію для заданої формули. Порівняти якість побудованих моделей. Визначити оптимальну модель. Вибір оптимальної моделі обґрунтувати.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв'язання задач.
5. Висновки.

Контрольні запитання

1. Поясніть призначення функції tree
2. Назвіть основні параметри функції tree
3. Поясніть призначення функції snip.tree
4. Назвіть основні параметри функції snip.tree
5. Поясніть призначення функції prune.tree
6. Назвіть основні параметри функції prune.tree

ЛАБОРАТОРНА РОБОТА № 7

Тема: Метод опорних векторів

Мета: Навчитися виконувати класифікацію об’єктів за допомогою вбудованих функцій системи R

Теоретичні відомості

Метод опорних векторів – це набір схожих алгоритмів виду «навчання із вчителем». Ці алгоритми зазвичай використовуються для задач класифікації та регресійного аналізу. Метод належить до розряду лінійних класифікаторів. Також може розглядатись як особливий випадок регуляризації за А. Н. Тихоновим. Особливою властивістю методу опорних векторів є безперервне зменшення емпіричної помилки класифікації та збільшення проміжку. Тому цей метод також відомий як метод класифікатора з максимальним проміжком. Основна ідея методу опорних векторів – перевід вихідних векторів у простір більш високої розмірності та пошук роздільної гіперплощини з максимальним проміжком у цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє наші класи. Роздільною гіперплощиною буде та, що максимізує відстань до двох паралельних гіперплощин. Алгоритм працює у припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим меншою буде середня помилка класифікатора.

Для класифікації об’єктів методом опорних векторів в системі R існує функція ***svm***.

Функція ***svm***

Опис функції

svm(formula, data= NULL, ..., subset, na.action = na.omit, scale = TRUE)
svm(x, y = NULL, scale = TRUE, type = NULL, kernel = "radial", degree = 3, gamma = if (is.vector(x)) 1 else 1/ ncol(x),
coef0 = 0, cost = 1, nu = 0.5, class.weights = NULL, cachesize = 40,
tolerance = 0.001, epsilon = 0.1, shrinking = TRUE, cross = 0, probabilit = FALSE, fitted = TRUE, ..., subset, na.action = na.omit)

де

formula – символьний опис моделі;

data – додатковий набір даних змінних моделей;

x – матриця, вектор або розріджена матриця (об'єкт класу ***matrix.csr***, що реалізований в пакеті ***SparseM***) об'єктів;

y – вектор відповідей для кожного рядка ***x***. Може бути вектором категорій (для задачі класифікації) або числовим вектором (для регресії);

scale – булевий вектор. Якщо довжина вектора ***scale*** становить 1, то його значення поширюється на всі змінні. За промовчуванням, дані масштабуються всередині функції таким чином, щоб математичне сподівання дорівнювало нулю, дисперсія – 1. Значення, які відповідають зсуву та коефіцієнту масштабування повертаються та можуть бути використані для прогнозування;

type – svm-тип розв'язуваної задачі:

– *C-classification;*

– *nu-classification;*

– *one-classification (for novelty detection);*

– *eps-regression;*

– *nu-regression;*

kernel – тип ядра, що використовується для навчання та прогнозування:

– ***linear*** (u, v) – лінійне;

– ***polynomial*** $(gamma \cdot (u,v) + coef\ 0)^{\deg ree}$ – поліноміальне;

– ***radial basis*** $e^{-\mathrm{gamma}}\|u - v\|^2$– радіально-базисне;

– ***sigmoid tanh***$(gamma \cdot (u,v) + coef\,0)$ – сигмоїдне;

degree – необхідний для ядра типу ***polynomial*** параметр (за промовчуванням дорівнює 3);

gamma – параметр, необхідний для всіх типів ядер, крім ***linear*** (за промовчуванням дорівнює 1/(розмірність даних));

coef 0 – параметр необхідний для ядер типу ***polynomial*** та ***sigmoid*** (за промовчуванням дорівнює 0);

cost – ціна невиконання обмежень (за промовчуванням дорівнює 1). Відповідає константі C в методі C-класифікації;

nu – параметр необхідний при використанні ***nu-classification, one-classification*** та ***nu-regression***;

class.weights – вектор вагових коефіцієнтів різних класів з назвою (використовується у випадку, коли розміри класів істотно відрізняються). Допускається, що деякі вагові множники можуть бути не вказані (за

промовчуванням встановлене значення вагового множника дорівнює 1). Обов'язково всі компоненти вектора повинні мати назву.

cachesize – розмір кешу (за промовчуванням 40 Мб);

tolerance – допустима точність розв'язку в критерії завершення роботи методу (за промовчуванням 0,001);

epsilon – значення ε у функції чутливості (***insensitive-loss function***) (за промовчування 0,1);

shrinking – використання евристики стиснення (***shrinking***) (за промовчуванням ***TRUE***);

cross – оцінка якості моделі шляхом виконання k-кратної перехресної перевірки при встановленні значення значення k > 0;

fitted – адаптація значення змінних (за промовчуванням TRUE);

probability – ймовірнісне прогнозування;

… – додаткові параметри для низькорівневої функції ***svm.default***;

subset – індексний вектор, який визначає прецеденти, які необхідно використати в навчальній вибірці;

na.action – функція, що визначає дії, які необхідно виконати при виявленні пропущених значень (***NA***). Якщо встановлено значення ***na.omit***, то будуть вилучені прецеденти з пропущеними значеннями необхідних змінних. При встановленні ***na.fail***, виявивши пропущені значення система повідомить про помилку.

Результатом роботи функції є об'єкт класу 'svm', який містить навчену модель з полями:

SV – опорні вектори;

index – індексний вектор, який визначає опорні вектори у матриці початкових даних. Індексами позначають вже оброблені початкові дані (після використання ***na.omit*** та ***subset***);

coefs – відповідні коефіцієнти помножені на навчальні ваги;

rho – від'ємний інтервал;

sigma – якщо регресійна модель є ймовірнісною, то параметр масштабування гіпотетичного (1-0) розподілу Лапласа, оцінений за допомогою максимальної правдоподібності;

probA, probB – числовий вектор довжини k(k -1)/2 (k - число класів), який містить параметри логістичних розподілів.

Приклад 1. Дослідити навчену модель, використовуючи ядро типу ***linear***. Для всіх інших параметрів використовувати встановлені за промовчуванням значення.

Розв'язання:

Крок 1. Встановити бібліотеку ***e1071***. Для цього перейти до меню ***Packages*** головного меню системи R. У списку, що з'явиться обрати ***Install package(s)***. У вікні ***CRAN*** вказати локальну належність та натиснути кнопку ***OK***. Після виконання описаних дій на екрані з'явиться вікно ***Packages*** з переліком бібліотек, які можна встановити. Клікнути назву бібліотеки ***e1071*** та натиснути кнопку ***OK***.

Крок 2. Ініціалізувати бібліотеку ***e1071.***

```
> library(e1071)
```

Крок 3. Завантажити навчальну вибірку в ***data.frame SVMData*** з файлу svmdata.txt**.**

```
> SVMData=read.table("svmdata.txt")
```

Крок 4. Виконати класифікацію за допомогою функції ***svm*** з ядром ***linear*** та значенням константи ***C***, що дорівнює **2**.

Крок 5. Виконати графічну інтерпретацію результатів навчання.

```
> plot(model,SVMData)
```

Крок 6. Обчислити число помилок на навчальній вибірці.

```
> x=subset(SVMData,select=-Color)
```

Крок 7. Виконати прогнозування за допомогою функції ***predict*** на навчальній моделі.

```
> Color_pred=predict(model,x)
```

Крок 8. Відобразити число помилок прогнозування за допомогою функції ***table***.

```
> table(SVMData$Color,Color_pred)
```

Крок 9. Оцінити якість навчання за числом помилок на тестовій вибірці.

```
> SVMDataTest <- read.table("svmdatatest.txt")
> x <- subset(SVMDataTest, select = -Color)
> Color_pred <- predict(model, x)
> table(SVMDataTest$Color,  Color_pred)
> model=svm(SVMData$Color~.,SVMData,kernel="linear",cost=2)
```

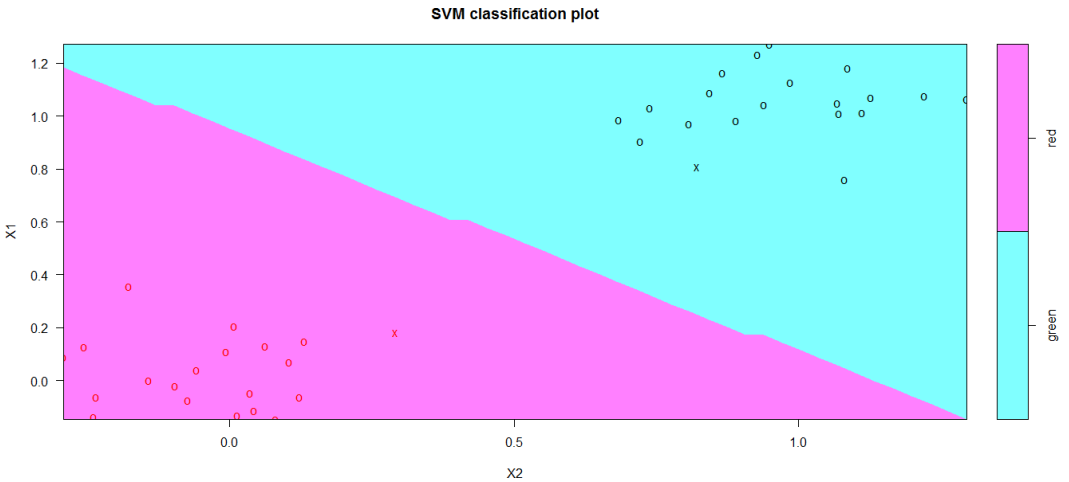

Рис. 14. Графічна інтерпретація результатів класифікації

Завдання

Використовуючи тестові вибірки, які знаходяться у файлах для аналізу ***svmdata.txt*** та ***svmdatatest.txt*** з папки «Файли для аналізу».

1. Дослідити навчену модель, використовуючи ядро типу ***linear***. Для всіх інших параметрів використовувати встановлені за промовчуванням значення.

2. Використовуючи ядро типу ***linear*** та змінюючи значення константи ***C*** досягти:

– нульової кількості помилок класифікації на навчальній вибірці;

– нульової кількості помилок класифікації на тестовій вибірці.

Визначити оптимальнезначення константи, вибір обґрунтувати. Чи завжди потрібно мінімізувати помилку на навчальній вибірці?

3. Визначити яке з ядер ***polynomial, radial*** чи ***sigmoid*** забезпечує мінімальну кількість помилок на тестовій вибірці при заданих значеннях параметра ***degree***.

4. Визначити яке з ядер ***polynomial***, ***radial*** чи ***sigmoid*** забезпечує мінімальну кількість помилок на тестовій вибірці.

5. Визначити яке з ядер ***polynomial***, ***radial*** чи ***sigmoid*** забезпечує мінімальну кількість помилок на тестовій вибірці. Експериментально пояснити як залежить ефективність перенавчання моделі від значення параметра ***gamma***.

6. Дослідити як залежить значення середньої квадратичної помилки рівняння регресії від значення параметра ε навчальної вибірки.

Зміст звіту

1. Тема та мета роботи
2. Короткі теоретичні відомості
3. Завдання до роботи згідно варіанту
4. Протоколи розв’язання задач
5. Висновки

Контрольні запитання

1. Поясніть суть методики «розділяй та володарюй»
2. Поясність суть алгоритму ID3
3. Поясність суть алгоритму C4.5
4. Поясність суть алгоритму CART
5. Поясність суть алгоритму методу опорних векторів
6. Поясність суть методу «найближчого сусіда»

ЛАБОРАТОРНА РОБОТА № 8

Тема: Використання бустинг алгоритму в задачах класифікації

Мета: Навчитися розв'язувати задачі класифікації за допомогою вбудованих функцій системи R

Теоретичні відомості

Набір моделей, які спільно використовуються для розв'язання однієї задачі називається **ансамблем моделей**.

Для формування ансамблів моделей використовуються методи бегінінгу, бустингу та стекінгу.

Бегінг формує набір класифікаторів, які комбінуються шляхом голосування або усереднення.

На відміну від бегінга бустинг є дещо складнішою, але ефективнішою процедурою. Подібно бегінгу бустинг використовує нестійкість алгоритмів навчання та створює ансамблі на базі єдиної вхідної множини. Але якщо в бегінгу моделі будуються паралельно та незалежно одна від іншої, то в бустингу кожна нова модель будується на базі результатів, що були отримані раніше, тобто моделі створюються послідовно. Бустинг створює моделі таким чином, щоб вони доповнювали раніше побудовані моделі, виконували ту роботу, яку інші моделі на попередніх кроках виконати не могли. І нарешті, остання відмінність бустингу від бекінга полягає в тому, що всім побудованим моделям залежно від їх точності присвоюються вагові множники.

Для побудови ансамблів моделей в системі R існують функції ***bagging*** та ***adaboost.M1***, в яких відповідно реалізовані методи бегенінгу та бустингу.

Функція ***adaboost.M1***

Опис функції

adaboost.M1(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman', minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))

де

formula – символьний опис моделі;

data – набір даних;

boos – якщо набуває значення ***TRUE*** (за промовчуванням), то використовуючи ваги прецедентів на поточній ітерації, з тренувальної множини генерується ***bootstrap*** набір даних;

mfinal – число ітерацій роботи алгоритму бустингу (число дерев рішень, які необхідно побудувати). За промовчуванням число ітерацій ***mfinal=100***;

coeflearn – якщо ***'Breiman'*** (за промовчуванням), то використовується ***alpha=1/2ln((1-err)/err).*** Якщо ***Freund***, то використовується ***alpha=ln((1-err)/err)***, де ***alpha*** – коефіцієнт оновлення ваг прецедентів.

minsplit – мінімально необхідно число прецедентів вершини, необхідне для роботи алгоритму поділу (спліт);

cp – параметр складності. Поділ, який не зменшує ***the overall lack of fit*** на величину ***cp***, не застосовується;

maxdepth – максимальна глибина будь-якої вершини в останньому дереві (глибина кореневої вершини вважається рівною 0), яка за промовчуванням дорівнює числу кластерів.

повертає список, який містить поля:

formula – використаний символьний опис моделі;

trees – набір дерев, які використовувалися при побудові моделей;

weights – вектор вагових множників дерев на всіх ітераціях;

votes – матриця, яка місить інформацію про кількість дерев для кожного претендента (вага кожного дерева враховується з урахуванням коефіцієнта ***alpha***), які «проголосували» за приналежність прецедента до того чи іншого класу;

class – прогнозований ансамблем класифікаторів клас прецедента;

importance – відносна «важливість» кожної змінної в задачі класифікації (скільки разів кожна змінна вибиралась для побудови спліта).

Функція ***bagging***

Опис функції

bagging(formula, data, mfinal = 100, minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))

де

formula – символьний опис моделі;

data – набір даних;

mfinal – число ітерацій роботи алгоритму бустингу (число дерев рішень, які необхідно побудувати). За промовчуванням число ітерацій ***mfinal=100***;

minsplit – мінімально необхідно число прецедентів вершини, необхідне для роботи алгоритму поділу (спліт);

cp – параметр складності. Поділ, який не зменшує ***the overall lack of fit*** на величину ***cp***, не застосовується;

maxdepth – максимальна глибина будь-якої вершини в останньому дереві (глибина кореневої вершини вважається рівною 0), яка за промовчуванням дорівнює числу кластерів.

повертає список, який містить поля:

formula – використаний символьний опис моделі;

trees – набір дерев, які використовувалися при побудові моделей;

votes – матриця, яка місить інформацію про кількість дерев для кожного претендента (вага кожного дерева враховується з урахуванням коефіцієнта ***alpha***), які «проголосували» за приналежність прецедента до того чи іншого класу;

samples – бутстрап набори даних (для всіх ітерацій).

class – прогнозований ансамблем класифікаторів клас прецедент;

importance – відносна «важливість» кожної змінної в задачі класифікації (скільки разів кожна змінна вибиралась для побудови спліта).

Приклад 1. Дослідити залежність тестової помилки від кількості дерев в ансамблі за допомогою алгоритму ***adaboost.M1*** для набору даних ***Vehicle*** з пакету (навчальна вибірка повинна складатися з 2/3*l прецедентів, які містяться в наборі даних). Побудувати графік залежності тестової помилки від кількості дерев, число яких може становити 11,71, 131, 211, 301. Пояснити отримані результати.

Розв'язання:

Крок 1. Встановити бібліотеку rpart. Для цього перейти до меню **Packages** головного меню системи R. У списку, що з'явиться обрати **Install package(s)**. У вікні **CRAN** вказати локальну належність та натиснути кнопку **ОК**. Після виконання описаних дій на екрані з'явиться вікно **Packages** з переліком бібліотек, які можна встановити. Клікнути назву бібліотеки **rpart** та натиснути кнопку **ОК**.

Крок 2. Встановити бібліотеку **mlbench.** ***Крок 3.*** Встановити бібліотеку **adabag.** ***Крок 4.*** Завантажити бібліотеку rpart.

```
library(rpart)
```

Крок 5. Завантажити бібліотеку mlbench.

```
library(mlbench)
```

Крок 6. Завантажити бібліотеку adabag.

```
library(adabag)
```

Крок 7. Завантажити набір даних Vehicle.

```
data(Vehicle)
```

Крок 8. Згенерувати вектор, що містить номери прецидентів, з яких буде сформовано навчальну вибірку (їх кількість становить 2/3**l*, где *l* – число прецидентів в наборі даних Vehicle).

```
l <- length(Vehicle[,1])
sub <- sample(1:l,2*l/3)
```

Крок 9. Встановити максимальне число ітерацій роботи алгоритму adaboost.M1. Нехай максимальне число ітерацій роботи алгоритму становить 25.

```
mfinal <- 25
```

Крок 10. Встановити максимальну глибину кожного дерева рішень в алгоритмі adaboost.M1. Нехай максимальна глибина дерева рішень становить 5.

```
maxdepth <- 5
```

Крок 11. Побудувати просте дерево рішень на підмножини прецидентів набору даних Vehicle з номерами з sub:

```
Vehicle.rpart <- rpart(Class~.,data=Vehicle[sub,],maxdepth=maxdepth)
```

Крок 12. Використовуючи побудовану модель, спрогнозувати відповіді на прецедентах для набору даних Vehicle з номерами, які не ввійшли до sub:

```
Vehicle.rpart.pred <- predict(Vehicle.rpart,newdata=Vehicle[-sub, ],type="class")
```

Крок 13. Обчислити помилку прогнозування

```
tb <- table(Vehicle.rpart.pred,Vehicle$Class[-sub])
 > error.rpart <- 1-(sum(diag(tb))/sum(tb))
 > error.rpart
       [1] 0.3687943
```

Крок 14. Побудувати ансамбль дерев рішень на підмножині прецидентів набору даних Vehicle з номерами з sub, використовуючи adaboost.M1:

```
> Vehicle.adaboost1 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=11,
coeflearn="Zhu", control=rpart.control (maxdepth=maxdepth))
> Vehicle.adaboost2 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=71,
coeflearn="Zhu", control=rpart.control (maxdepth=maxdepth))
> Vehicle.adaboost3 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=131,
coeflearn="Zhu", control=rpart.control (maxdepth=maxdepth))
> Vehicle.adaboost4 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=211,
coeflearn="Zhu", control=rpart.control (maxdepth=maxdepth))
> Vehicle.adaboost5 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=301,
coeflearn="Zhu", control=rpart.control (maxdepth=maxdepth))
> Vehicle.adaboost5 <- boosting(Class ~.,data=Vehicle[sub, ], mfinal=301,
coeflearn="Zhu", control=rpart.control
```

Крок 15. Використовуючи побудовану модель, спрогнозуємо відповіді на прецидентах для набору даних Vehicle з номерами, які не ввійшли до sub:

```
> Vehicle.adaboost.pred1 <- predict.boosting(Vehicle.adaboost1,newdata=Vehicle[-sub, ])
> Vehicle.adaboost.pred2 <- predict.boosting(Vehicle.adaboost2,newdata=Vehicle[-sub, ])
> Vehicle.adaboost.pred3 <- predict.boosting(Vehicle.adaboost3,newdata=Vehicle[-sub, ])
> Vehicle.adaboost.pred4 <- predict.boosting(Vehicle.adaboost4,newdata=Vehicle[-sub, ])
> Vehicle.adaboost.pred5 <- predict.boosting(Vehicle.adaboost5,newdata=Vehicle[-sub, ])
> Vehicle.adaboost.pred1[-1]
> error.rpart
```

Крок 16. Побудувати графік залежності тестової помилки від кількості дерев.

```
> y=c(Vehicle.adaboost.pred1$error, Vehicle.adaboost.pred2$error,
Vehicle.adaboost.pred3$error, Vehicle.adaboost.pred4$error,
Vehicle.adaboost.pred5$error)
> x=c(11,71,131,211,301)

> plot(x,y,type="l")
```

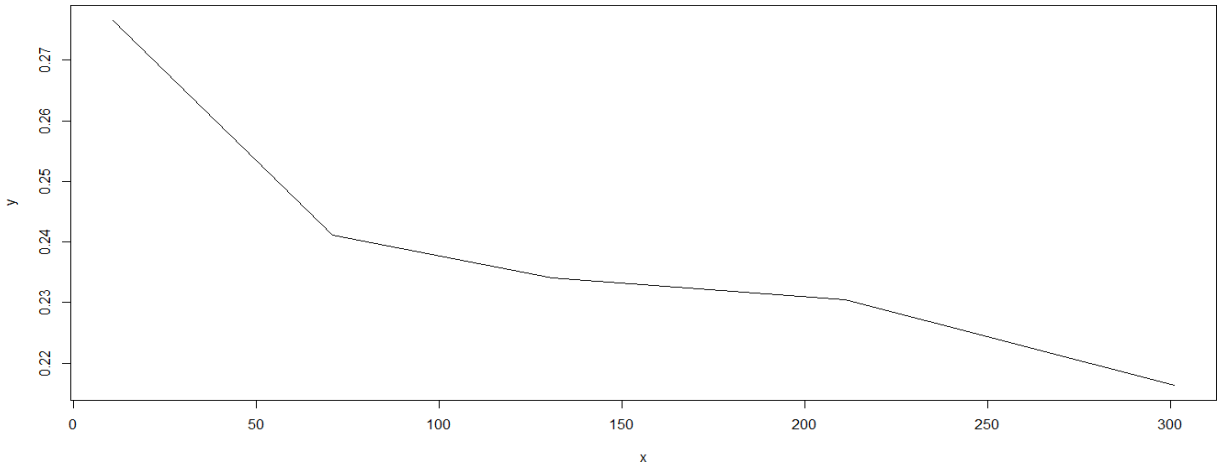

Рис. 15. Графік залежності тесової помилки від кількості дерев ансамблю

Найменша тестова похибка спостерігається при N=300, а найбільша при N=11.

Приклад 2. Дослідити залежність тестової помилки від кількості дерев в ансамблі за допомогою алгоритму ***bagging*** для набору даних ***Glass*** з пакету ***mlbench*** (навчальна вибірка повинна складатися з 2/3* l прецедентів, які містяться в наборі даних). Побудувати графік залежності тестової помилки від кількості дерев, число яких може становити 11,51,111,171,201. Пояснити отримані результати.

Розв'язання:

Крок 1. Встановити бібліотеку ***rpart*** (якщо вона не встановлена). Для цього перейти до меню ***Packages*** головного меню системи R. У списку, що з'явиться обрати ***Install package(s)***. У вікні ***CRAN*** вказати локальну належність та натиснути кнопку **ОК**. Після виконання описаних дій на екрані з'явиться вікно ***Packages*** з переліком бібліотек, які можна встановити. Клікнути назву бібліотеки ***rpart*** та натиснути кнопку ***ОК***.

Крок 2. Встановити бібліотеку ***mlbench*** (якщо вона не встановлена).

Крок 3. Встановити бібліотеку ***adabag*** (якщо вона не встановлена)**.**

Крок 4. Завантажити бібліотеку ***rpart***.

```
>library(rpart)
```

Крок 5. Завантажити бібліотеку ***mlbench***.

```
>library(mlbench)
```

Крок 6. Завантажити бібліотеку ***adabag***.

```
>library(adabag)
```

Крок 7. Завантажити набір даних ***Glass***.

```
>data(Glass)
```

Крок 8. Згенерувати вектор, що містить номери прецидентів, з яких буде сформовано навчальну вибірку (їх кількість становить 2/3*l, где l – число прецидентів в наборі даних ***Glass***).

```
>l <- length(Glass [,1])
>sub <- sample(1:l,2*l/3)
```

Крок 9. Встановити максимальне число ітерацій роботи алгоритму. Нехай максимальне число ітерацій роботи алгоритму становить 51.

```
>mfinal <- 51
```

Крок 10. Встановити максимальну глибину кожного дерева рішень. Нехай максимальна глибина дерева рішень становить 5.

```
>maxdepth <- 5
```

Крок 11. Побудувати просте дерево рішень на підмножини прецидентів набору даних ***Glass*** з номерами з ***sub***:

```
>  Glass.rpart <- rpart(Type~.,data= Glass[sub,],maxdepth=maxdepth)
```

Крок 12. Використовуючи побудовану модель, спрогнозувати відповіді на прецедентах для набору даних ***Glass*** з номерами, які не ввійшли до ***sub***:

```
> Glass.rpart.pred <- predict(Glass.rpart,newdata= Glass [-sub, ],type="class")
```

Крок 13. Обчислити помилку прогнозування

```
> tb <- table(Glass.rpart.pred, Glass $Type[-sub])
> error.rpart <- 1-(sum(diag(tb))/sum(tb))
> error.rpart
[1] 0.4027778
```

Крок 14. Побудувати ансамбль дерев рішень на підмножині прецидентів набору даних ***Glass*** з номерами з ***sub***, використовуючи ***bagging***:

```
> Glass.bagging1 <- bagging (Type ~.,data= Glass[sub, ], mfinal=11)
> Glass.bagging2 <- bagging (Type ~.,data= Glass[sub, ], mfinal=51)
> Glass.bagging3 <- bagging (Type ~.,data= Glass[sub, ], mfinal=111)
> Glass.bagging4 <- bagging (Type ~.,data= Glass[sub, ], mfinal=171)
> Glass.bagging5 <- bagging (Type ~.,data= Glass[sub, ], mfinal=201)
```

Крок 15. Використовуючи побудовану модель, спрогнозуємо відповіді на прецидентах для набору даних ***Glass*** з номерами, які не ввійшли до ***sub***:

```
> Glass.bagging.pred1 <- predict.bagging(Glass.bagging1,newdata= Glass[-sub, ])
> Glass.bagging.pred2 <- predict.bagging(Glass.bagging2,newdata= Glass[-sub, ])
> Glass.bagging.pred3 <- predict.bagging(Glass.bagging3,newdata= Glass[-sub, ])
> Glass.bagging.pred4 <- predict.bagging(Glass.bagging4,newdata= Glass[-sub, ])
> Glass.bagging.pred5 <- predict.bagging(Glass.bagging5,newdata= Glass[-sub, ])
> Glass.bagging.pred1[-1]
> error.rpart
```

Крок 16. Побудувати графік залежності тестової помилки від кількості дерев.

```
> z=c(Glass.bagging.pred1$error, Glass.bagging.pred2$error, Glass.bagging.pred3$error, Glass.bagging.pred4$error, Glass.bagging.pred5$error)
> xx=c(11,51,111,171,201)
> plot(xx,y,type="l")
```

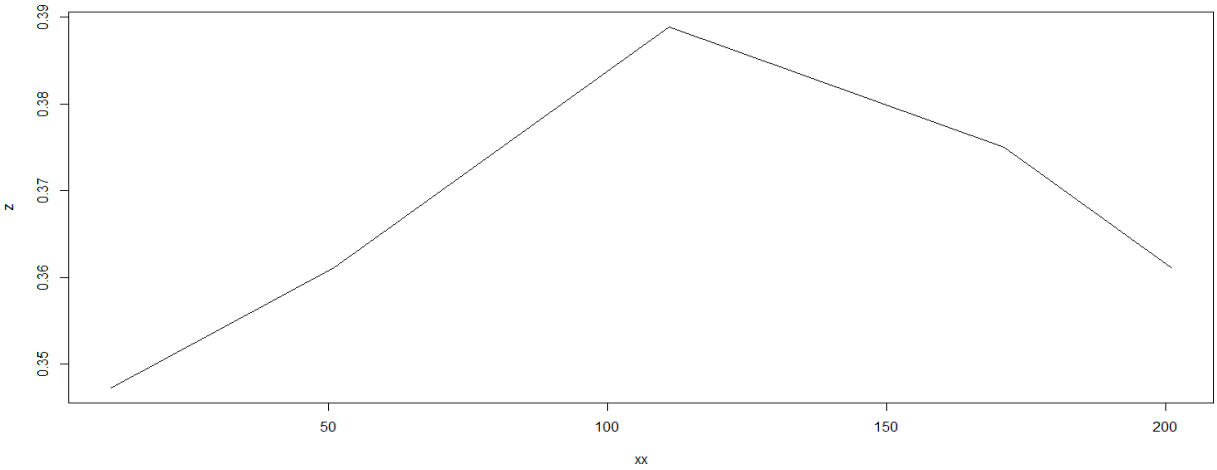

Рис. 16. Графік залежності тесової помилки від кількості дерев ансамблю

Завдання

1. Дослідити залежність тестової помилки від кількості дерев в ансамблі за допомогою алгоритму ***adaboost.M1*** для набору даних ***Vehicle*** з пакету ***mlbench*** (навчальна вибірка повинна складатися з 7/10 прецедентів, які містяться в наборі даних). Побудувати графік залежності тестової помилки від кількості дерев, число яких може становити 1,11,21,….301. Пояснити отримані результати.

2. Дослідити залежність тестової помилки від кількості дерев в ансамблі за допомогою алгоритму ***bagging*** для набору даних ***Glass*** з пакету ***mlbench*** (навчальна вибірка повинна складатися з 7/10 прецедентів, які містяться в наборі даних). Побудувати графік залежності тестової помилки від кількості дерев, число яких може становити 1,11,21,….301. Пояснити отримані результати.

3. Реалізувати бустинг алгоритм з трьома класифікаторами. Порівняти тестову помилку, отриману з використанням даного класифікатора на наборі даних ***Vehicle*** і ***Glass*** з тестовою помилкою, отриманою з використанням одиничного дерева класифікації.

Зміст звіту

1. Тема та мета роботи
2. Короткі теоретичні відомості
3. Завдання до роботи згідно варіанту
4. Протоколи розв’язання задач
5. Висновки

Контрольні запитання

1. Поясніть призначення функції ***adaboost.M1***
2. Назвіть основні параметри функції ***adaboost.M1***
3. Назвіть основні параметри функції ***bagging***
4. Поясніть призначення функції ***bagging***

ЛАБОРАТОРНА РОБОТА № 9

Тема: Дослідження вибірки за допомогою функції bootstrap

Мета: Навчитися досліджувати вибірки за допомогою вбудованих функцій системи R

Теоретичні відомості

Функція *bootstrap*

Опис функції

bootstrap(x,nboot,theta,..., func=NULL)

де

x – вектор набору даних;

nboot – кількість наборів даних;

theta – додатковий параметр;

... – додаткові аргументи, які передаються функції ***theta***;

func – функція розподілу $\hat{\theta}$. Якщо аргумент ***func*** визначений, то повертається значення стандартної помилки для цієї величини.

Результатом роботи функції є модель з полями:

thetastar – nboot bootstrap значень ***theta;***

func.thetastar – функціонал ***func bootstrap*** розподілу ***thetastar***, якщо ***func*** визначено;

jack.boot.val – ***the jackknife-after-bootstrap*** значення для ***func***, якщо ***func*** визначено;

jack.boot.se – the jackknife-after-bootstrap стандартна помилка ***func***, якщо ***func*** визначено;

call – рядок «розподіленого» звертання до функції.

Приклад 1. Згенерувати вектор довжини $N=10$ нормально розподіленої випадкової величини $N(0,1)$. Обчислити стандартну помилку для статистичного математичного сподівання, використовуючи ***bootstrap*** з числом повторів $B=100$ та порівняти з теоретичною оцінкою.

Розв'язання:

Крок 1. Встановити бібліотеку ***bootstrap***. Для цього перейти до меню ***Packages*** головного меню системи R. У списку, що з'явиться обрати ***Install package(s).*** У вікні ***CRAN*** вказати локальну належність та натиснути кнопку ***OK***. Після виконання описаних дій на екрані з'явиться вікно ***Packages*** з переліком бібліотек, які можна встановити. Клікнути назву бібліотеки ***bootstrap*** та натиснути кнопку ***OK***.
Крок 2. Ініціалізувати бібліотеку ***bootstrap.***
>library(bootstrap)
Крок 3. Згенерувати вектор нормально розподілених випадкових величин,що містить 10 елементів.
>v=runif(10)
Крок 4. Обчислити 0,05-квантиль роподілу
>perc05 <- function(v){quantile(v, .05)}

Крок 5. Виконати процедуру ***bootstrap.***
>results <- bootstrap(v, 100, perc05)

Крок 6. Обчислити математичне сподівання для отриманих значень bootstrap-оцінок. Для цього присвоїти параметру func значення вбудованої функції mean.
> results <- bootstrap(v, 100, perc05, func=mean)?

Стандартна помилка для статистичного математичного сподівання розрахована за допомогою функції ***bootstrap*** з числом повторів B=100 дорівнює 0.03747417.

Функція *boott*

Опис функції

boott(x,theta, ..., sdfun=sdfunboot, nbootsd=25, nboott=200, VS=FALSE, v.nbootg=100, v.nbootsd=25, v.nboott=200, perc=c(.001,.01,.025,.05,.10,.50,.90,.95,.975,.99,.999))

де

x – вектор набору даних;

theta – додатковий параметр;

... – додаткові аргументи, які передаються функції ***theta***;

sdfun – функція обчислення стандартної помилки для ***theta***. Набуває вигляду ***sdmean=function(x,nbootsd,theta,...)***, де ***nbootsd*** – аргумент, який не використовується. Якщо ***theta*** – математичне сподівання, то, наприклад, ***sdmean=function(x,nbootsd,theta,...) {sqrt(var(x)/length(x))}.*** Якщо значення ***sdfun*** не задано, то ***boott*** використовує внутрішній ***bootstrap*** цикл, щоб оцінити стандартну помилку для ***theta(x)***;

nbootsd – кількість число ***bootstrap samples***, які використовуються для оцінки стандартної помилки для ***theta(x)***;

nboott – кількість ***bootstrap samples***, які використовуються для обчислення значення ***bootstrap*** T-статистики. Значення абсолютного мінімуму становить 200. Для достовірної α% довірчої точки необхідно, щоб це значення встановило більше або дорівнювало 1000. Загальне число ***bootstrap samples*** дорівнює ***nboot*nbootsd***;

VS – якщо цей параметр дорівнює ***TRUE***, то обчислюється значення перетворення для стабілізації дисперсії; довірчий інтервал визначається у перетвореному просторі. Якщо параметр дорівнює ***FALSE***, то стабілізація дисперсії не виконується;

v.nbootg – кількість ***bootstrap samples***, які використовуються для обчислення перетворення g, яке стабілізує дисперсію. Використовується у тих випадках, коли ***VS=TRUE***;

v.nbootsd – число ***bootstrap samples,*** які використовуються для обчислення стандартного відхилення ***theta(x).*** Використовується у тих випадках, якщо ***VS=TRUE***;

v.nboott – кількість ***bootstrap samples***, які використовуються для обчислення bootstrap T-статистики. Використовуються у випадку, якщо ***VS=TRUE***. Загальна кількість ***bootstrap samples*** дорівнює ***v.nbootg*v.nbootsd + v.nboott***;

perc – довірчі точки.

Результатом роботи функції є модель з полями:

confpoints – обчислене значення довірчих точок;

theta, g – значення ***theta*** і ***g*** повертаються, якщо ***VS=TRUE***. ***(theta[i],g[i]), i=1,length(theta)*** – це значення функції перетворення ***g***, яка стабілізує дисперсію, в точках ***theta[i].***

call – рядок «розподіленого» звертання до функції.

Приклад 2. Побудувати довірчі інтервали для математичного сподівання множини даних, що розподілені за експоненційним законом.

Розв'язання:

Крок 1. Ініціалізувати бібліотеку **bootstrap.**

```
>library(bootstrap)
```

Крок 2. Згенерувати вектор експоненційно розподілених випадкових величин з параметром □ □ 2 , що містить 20 елементів.

```
>x <- rexp(1000, rate = 2)
```

Крок 3. Обчислити стандартну помилку для theta(x):

```
>sdmean <- function(x,nbootsd,theta){sqrt(var(x)/length(x))}
```

Крок 4. Виконати процедуру boott для обчислення 0.05 и 0.95 confidence points с використанням перетворення, яке стабілізує дисперсію**:**

```
>results <- boott(x, mean, sdfun=sdmean, VS=TRUE, perc=c(0.05,0.95), nboott = 2000)
```

Функція *abcnon*

Опис функції

abcnon(x, tt, epsilon=0.001, alpha=c(0.025, 0.05, 0.1, 0.16, 0.84, 0.9, 0.95, 0.975))

де

x – вектор або матриця початкових даних;

tt – функція, яка визначає параметр ***resamling form tt(p,x)***, де ***p*** – вектор відношень, ***x*** – величина кроку в методі скінчених різниць;

alpha – довірчий інтервал.

Результатом роботи функції є модель з полями:

limits – оцінка ***confidence points***, отримана за допомогою ***ABC*** та стандартних нормальних методів;

stats – список, який містить ***t0*** – спостережуване значення ***tt, sighat – infinitesimal jackknife*** стандартну помилку для ***tt, bhat*** – оціночний ***bias; constants*** – список, який містить ***a*** – константу прискорення, ***z0*** – регулювання, ***bias, cq*** – компонента спотворення;

tt.inf – апроксимовані коефіцієнти впливу для ***tt***;

pp – матриця, рядки якої є ***resampling points*** з найменшим степенем відповідності.

The abc confidence points – значення функції ***tt*** в даних точках.

Приклад 3. Побудувати довірчі інтервали для математичного сподівання множини даних, що розподілені за експоненційним законом.

Розв'язання:

Крок 1. Ініціалізувати бібліотеку ***bootstrap.***

```
>library(bootstrap)
```

Крок 2. Згенерувати вектор експоненційно розподілених випадкових величин з параметром $\lambda=2$, що містить 20 елементів.

```
>x <- rexp(1000, rate = 2)
```

Крок 3. Обчислити стандартну помилку для ***theta(x)*** як функцію від вектора даних x та вектора частоти потрапляння кожного елемента x_i в ***bootstrap samples*** – p:

```
>tt <- function(p,x) {sum(p*x)/sum(p)}
```

Крок 4. Виконати процедуру **abcnon** для обчислення 0.05 и 0.95 ***confidence points*** с використанням перетворення, яке стабілізує дисперсію**:**

```
> results<-abcnon(x, tt, alpha=c(0.05,0.95))
```

Завдання

1. Для ***N = 10,100,1000*** згенерувати вектор довжини ***N*** нормально ррозподіленої випадкової величини ***N (0,1)***. Обчислити стандартну помилку для статистичного математичного сподівання, використовуючи ***bootstrap*** з числом повторів ***B = 200***, дослідити її залежність від ***N***, порівняти з теоретичною оцінкою.

2. Завантажити набір даних ***mouse.t***, обчислити статистичне математичне сподівання та медіану. Оцінити стандартну помилку за допомогою функції ***bootstrap*** при ***B = 50,100,250,500,1000***.

3. Завантажити набір даних ***law*** та обчислити кореляцію між середнім балом, які набрали учні школи з тесту по інформатиці ***(LSAT)*** та їх середнім балом з усіх предметів ***(GPA)***. Оцінити стандартну помилку коефіцієнта кореляції за допомогою функції ***bootstrap***. Обчислити коефіцієнт кореляції на наборі даних ***law82*** та порівняти отримані результати.

4. Завантажити набір даних ***spatial***. Обчислити 90% інтервали довіри для статистичної дисперсії для ***A***, використовуючи функції 1) ***bootstrap-t***, 2) ***BCα***, 3) ***ABC***, 4) ***bootstrap percentile***, та порівняйте отримані результати.

Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Завдання до роботи згідно з варіантом.
4. Протоколи розв’язання задач.
5. Висновки.

Контрольні запитання

1. Поясніть призначення функції ***bootpred***
2. Назвіть основні параметри функції ***bootpred***
3. Поясніть призначення функції ***bcanon***
4. Назвіть основні параметри функції ***bcanon***
5. Поясніть призначення функції ***abcpar***
6. Назвіть основні параметри функції ***abcpar***
7. Поясніть призначення функції ***abcnon***
8. Назвіть основні параметри функції ***abcnon***
9. Поясніть призначення функції ***boott***
10. Назвіть основні параметри функції ***boott***
11. Поясніть призначення функції ***bootstrap***

Список літератури

1. М.В. Талах, В.В. Дворжак Інтелектуальний аналіз даних. Частина 1 / М.В. Талах, В.В. Дворжак – Чернівці: Технодрук, 2022. – 367 с.

2. Гороховатський В.О., Творошенко І.С. Методи інтелектуального аналізу та оброблення даних: навч. посібник. – Харків: ХНУРЕ, 2021. – 92 с.

3. Іванов С.М., Максишко Н.К., Бречко Д.О. Інтелектуальний аналіз даних: конспект лекцій. Запоріжжя: ЗНУ, 2020, 156 с.

4. Болюбаш Н. М. Інтелектуальний аналіз даних : навч. посіб. / Н. М. Болюбаш. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 320 с

Додаткова:

5. Ліщинська Л. Б., Добровольська Н. В. Перспективні програмні інструменти для аналізу даних у бізнесі //Вісник Хмельницького національного університету Серія: «Технічні науки» №1, 2022. С.78-83 DOI: https://www.doi.org/10.31891/2307-5732-2022-305-1

6. Болюбаш Н. М. Методичні вказівки до виконання курсової роботи з дисципліни «Інтелектуальний аналіз» даних для студентів спеціальності 122 «Комп'ютерні науки» : методичні вказівки / Н. М. Болюбаш. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2021. – 28 с. – (Методична серія ; вип. 347). URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/478

7. Yanchang Zhao and Yonghua Cen (Eds.). Data Mining Applications with R.

8. K. Chupryna, I. Ivakhnenko, S. Biloshchytska, C. Mykhaylo, D. Ryzhakov and D. Sobol, "Formalized Management of Changes at the Enterprise by Means of Fuzzy Logic," 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, 2023, pp. 490-494, URL: https://doi.org/10.1109/SIST58284.2023.10223567

9. Biloshchytskyi, A., Omirbayev, S., Mukhatayev, A., Biloshchytska, S., Toxanov, S., & Faizullin, A. (2024). The concept of the Internet of Things in the development of information and analytical systems based on the method of constructing a scalar assessment of the results of research activities of scientists. Procedia Computer Science, 231, 684–690. URL: https://doi.org/10.1016/j.procs.2023.12.161

10. Yehorov O., Volokh B., Bosenko I., Yeremenko B., Terenchuk S., Modeling of highly loaded distributed system supporting the process of assessing the technical condition of objects, CEUR Workshop Proceedings, 4th

International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2024, Ternopil, Ukraine, Opole, Poland, 23-25 October 2024, Vol. 3896, P. 173 – 185. URL: https://ceur-ws.org/Vol-3896/paper10.pdf

11. Босенко І.В. Моделі і методи штучного інтелекту в процесі виконання будівельно-технічної експертизи. Управління розвитком складних систем.Київ, 2025. №61. С. 180 – 186, https://doi.org/10.32347/2412-9933.2025.61.180-186.

Додаткові ресурси:

12. R: https://www.r-project.org/

13. The Comprehensive R Archive Network – https://cran.r-project.org/

14. RStudio - https://posit.co/download/rstudio-desktop/

15. Набори даних для аналізу: https://www.kaggle.com/datasets

Навчально-методичне видання

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Методичні вказівки
до виконання лабораторних робіт 1-9
для добувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 «Комп'ютерні науки»
та F6 «Інформаційні системи та технології

Укладачі: **Білощицька** Світлана Василівна,
Босенко Ігор Валерійович,
Мацієвський Олексій Олегович

Комп'ютерне верстання *А. П. Селівестрової*

Ум. друк. арк. 3,72. Обл.-вид. арк. 4,0
Електронний документ. Вид № 149/V-25

Виконавець і виготовлювач

Київський національний університет будівництва і архітектури
Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.